UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INFORMÁTICA

DOCTORADO EN INGENIERÍA INFORMÁTICA

Tecnologías Avanzadas de la información

TESIS DOCTORAL

Standard-Based Software Product Reliability Analysis

Doctorando: Felipe Febrero Hidalgo
Directora: Dra. María de los Ángeles Moraga de la Rubia
Directora: Dra. Coral Calero Muñoz

September, 2019
Abstract

Context: The increasing dependence of our society on software driven systems has led Software Product Quality to become a main concern as well as making it a highly active research area with hundreds of works being published every year. Software Product Quality is a complex multidimensional concept for which Reliability is considered as the key attribute. Notwithstanding, due to its conceptual complexity, there is no common agreement on what Software Reliability is, thus different stakeholders use a variety of Software Reliability views. Researches have also approached this topic from various and heterogeneous points of view, from the hardware legacy concept in terms of probability of failure to the recent proposals in terms of delivered service. This diversity is resulting in a rich body of literature on this topic the counterpart being its considerable complexity that makes it difficult to apprehend. Moreover, the sheer number of models proposed since the early 1970s has added an important confusion to a particularly complex body of knowledge. It would, however, appear that this research activity is much more reduced as regards how to apply representative International Standards on Software Quality to the Software Reliability assessment in real-world industrial environment, with just a few works on Standard-Based Software Reliability Modeling (SB-SRM). This is surprising given the relevance of such International Standards in industry.

Objective: Our main aim is to improve our understanding of what Software Reliability means for industrial stakeholders as well as that of contributing to enhance the industrial applicability of Software Quality Models. The objective of this research is therefore searching for an answer to the question of how to apply Software Product Reliability Modeling to industrial environments using representative International Standards as a basis. To accomplish that we have approaching Software Reliability analysis by using structural model based on representative International Standards, which are industry-oriented as well as searching to develop a straightforward to apply method, which is of paramount importance in real-world context.

Method: In order to achieve these stated objectives we first thoroughly performed a Systematic Review of the available literature, searching for confirmation or refutation of our research hypothesis as well as to constitute the State-of-the-Art on the topic. From this
research a model proposal emerged, which underwent a verification phase by peer-review and publication in representative journals as well as presented in International Conference. After verification the proposal was validated into real-world industrial context by means of an ad hoc Case Study which results are at the moment of writing these lines partially published.

**Results:** The obtained results showed a very limited application of SB-SRM particularly to industrial environment. The sensibleness of the proposed model and suitability of the analysis method is proved by the application on a very large industrial system which provides empirical evidence on the conceptual descriptiveness capturing stakeholders’ views and industrial applicability in an efficient manner. This application also proved its suitability as analysis tool within the continuous improvement process both, for quality managers and technical managers interested in the relationships between source code attributes and the system exhibited behaviour.

**Conclusion:** Our analysis points to the complexity of the proposed models together with the difficulties involved in applying them to the management of engineering activities as a root cause to be considered for such limited application. The various stakeholder needs are also an issue of paramount importance that should be better covered if the industrial applicability of the proposed models is to be increased. The industry needs of methods that are straightforward to apply in order to be useful to inform the decision making process in a real world context with a minimum need on resources and a maximum of results. This Thesis shows that structural models based on representative international standards can constitute a proper way to manage Software Product Quality in an environment where exhaustive testing is impractical and uneconomical.
Resumen

Contexto: La creciente dependencia de nuestra sociedad de sistemas gobernados mediante Software ha llevado a que la calidad del Producto Software se convierta en una preocupación mayor además de convertirla en un área de investigación de gran actividad con cientos de trabajos publicados cada año. La Calidad del Producto Software es un concepto complejo y multidimensional en el que la Fiabilidad se considera es el atributo clave. No obstante debido a su complejidad conceptual no existe consenso sobre qué es la Fiabilidad del Software, por lo que cada usuario del sistema mantiene un punto de vista diferente sobre la Fiabilidad del Software. La investigación académica también ha abordado este tema desde diferentes y heterogéneos puntos de vista, desde aquel heredado del campo de hardware en términos de probabilidad de fallo hasta las recientes propuestas en términos de servicio entregado. Esta diversidad está dando como resultado una gran cantidad de literatura sobre este tema, siendo su contrapartida una considerable complejidad que hace que sea difícil de aprehender. Además, la gran cantidad de modelos propuestos desde principios de la década de 1970 ha exacerbado la confusión en un conjunto de conocimientos que ya es por sí mismo particularmente complejo. Sin embargo parece que esta actividad de investigación es mucho más reducida en cuanto a cómo aplicar las Normas Internacionales de Calidad de Software a la evaluación de la Fiabilidad del Software en el entorno industrial, con solo unos pocos trabajos sobre Modelado de Fiabilidad de Software Basado en Estándares. Esto es sorprendente dada la relevancia de tales Estándares Internacionales en la industria.

Objetivo: Nuestro principal propósito es mejorar la comprensión de lo que significa la Fiabilidad del Software para las diferentes partes interesadas en él en un contexto industrial, así como contribuir a mejorar la aplicabilidad industrial de los Modelos de Calidad de Software. Por lo tanto, el objetivo de esta investigación es buscar una respuesta a la pregunta de cómo aplicar el Modelado de Fiabilidad del Producto Software a entornos industriales utilizando como base Estándares Internacionales ampliamente reconocidos. Para lograrlo, abordaremos el análisis de la Fiabilidad del Software mediante el uso de un modelo estructural basado en Estándares Internacionales, que están orientados a la industria, así como buscaremos desarrollar un método sencillo de aplicar, lo que es de suma importancia en el contexto del mundo real.
**Método:** A fin de lograr tales objetivos, primero realizamos una Revisión Sistemática de la literatura disponible, buscando la confirmación o refutación de nuestras hipótesis de investigación, así como también para constituir el Estado del Arte en el tema. De esta investigación surgió una propuesta de modelo que se sometió a una fase de verificación mediante revisión por pares y publicación en revistas representativas. Este modelo así como una primera aplicación fue también presentado en el marco de una Conferencia Internacional sobre Calidad y Fiabilidad de Software. Una vez superada la verificación, la propuesta fue validada en un contexto industrial mediante un Estudio de Caso ad hoc cuyos resultados se encuentran, en el momento de escribir estas líneas, también parcialmente publicados.

**Resultados:** Los resultados obtenidos mostraron una aplicación muy limitada del Modelado de Fiabilidad de Software Basado en Estándares, particularmente en el entorno industrial. La aplicación de la solución propuesta en esta Tesis a un sistema industrial muy grande muestra la sensatez del modelo propuesto y la idoneidad del método de análisis. De la misma manera proporciona una valiosa evidencia empírica respecto a su capacidad de descripción conceptual capturando los puntos de vista de los distintos interesados tanto como sobre la aplicabilidad industrial de manera eficiente. Esta aplicación también demostró su idoneidad como herramienta de análisis dentro del proceso de mejora continua tanto para gerentes de calidad como para gerentes técnicos interesados en las relaciones entre los atributos del código fuente y el comportamiento exhibido del sistema.

**Conclusión:** Nuestro análisis apunta a la complejidad de los modelos propuestos junto con las dificultades involucradas en su aplicación a la gestión de las actividades cotidianas de ingeniería como una causa raíz a considerar para una aplicación tan limitada. Las diversas necesidades de los interesados son también un punto de suma importancia que debe cubrirse mejor si perseguimos aumentar la aplicabilidad industrial de los modelos propuestos. La industria necesita métodos que sean fáciles de aplicar para ser útiles para informar el proceso de toma de decisiones en un contexto del mundo real con una necesidad mínima de recursos y un máximo de resultados. Esta Tesis muestra que los modelos estructurales basados en Estándares Internacionales pueden constituir una forma adecuada de gestionar la Calidad del Producto Software en un entorno en el cual las pruebas exhaustivas no son prácticas ni económicamente abordables.
# CONTENTS

List of Figures xi

List of Tables xiii

I Theoretical Basis and Proposal 1

1. Introduction 3
   1.1. Background and motivation 3
   1.2. Hypothesis and Objectives 7
   1.3. Sponsored Research Projects 8
   1.4. Thesis Structure 9

2. Research Methodologies 11
   2.1. Introduction 11
   2.2. Evidence Based Software Engineering 13
      2.2.1. Systematic Mapping Studies 16
      2.2.2. Systematic Literature review 18
      2.2.3. Thematic Synthesis 23
   2.3. Empirical Software Engineering 24
      2.3.1. Characteristics of Empirical Research 25
      2.3.2. The Case Study Methodology 28
      2.3.3. Empirical Research Validity 32
   2.4. Research Methods Applied in this Thesis 34

3. State-of-the-art 37
   3.1. Introduction 37
   3.2. SMS on Software Reliability Modelling 39
      3.2.1. Data gathering for the SMS 40
      3.2.2. Results of the SMS 46
      3.2.3. Answering the research questions 51
   3.3. The Systematic Literature Review on SB-SRM 55
      3.3.1. Data gathering for the SLR 58
      3.3.2. Results of the SLR 61
   3.4. Thematic Synthesis 63
   3.5. State-of-the-Art wrap-up 66
      3.5.1. Threats to validity 66
## 3.5.2. Conclusion

### 4. The Standard Based Model

#### 4.1. Standard Based Software Reliability Modelling

- **4.1.1. The need for Software Quality Models**
- **4.1.2. Industrial Relevance of International Standards**
- **4.1.3. Identified Hindrances**

#### 4.2. The Proposal

#### 4.3. The Industrial Application framework

- **4.3.1. Profitable Software Engineering**
- **4.3.2. The Human Factor**
- **4.3.3. The Incremental Approach to problem solving**

## II Application and Empirical Validation

### 5. Operationalization and Validation Strategy.

#### 5.1. Introduction

- **5.1.1. Aggregation Strategy**
- **5.1.2. Capturing Stakeholders view**
- **5.1.3. Failure Mechanism**
- **5.1.4. Software Usage Modelling**
- **5.1.5. The Case Study**

#### 5.2. The aggregation Strategies

- **5.2.1. The F-AHP fundamentals**
- **5.2.2. How to apply F-AHP In practice**

#### 5.3. From defects to Failures

- **5.3.1. The Software Failure Mechanism**
- **5.3.2. Defects Taxonomies**

#### 5.4. Modelling the System Usage

- **5.4.1. The Operational Profile**
- **5.4.2. Our proposed alternative**

#### 5.5. The Incremental Case Study strategy

### 6. The Incremental Case Study.

#### 6.1. The Case Study: Definition and Data Gathering

#### 6.2. The Case Study: Incremental Application and Analysis

- **6.2.1. The first incremental step: Proof-of-Concept and Feasibility**
- **6.2.2. The second incremental step: Users Perception of Reliability**
- **6.2.3. The third incremental step: From Attributes to Behavior**

#### 6.3. A Software Management Problem

#### 6.4. The Case Study: Empirical Validity and Conclusions

- **6.4.1. Empirical Validity**
- **6.4.2. Main Conclusions from the Case Study**

### 7. Research Outcomes.

#### 7.1. Analysis of Research Goals

#### 7.2. Support for Results

#### 7.3. Research Contributions
CONTENTS

7.4. Future Research Lines. ...................................................... 155

Appendices 157
A. Literature Review Search ................................................... 159
B. Software Product Defects Taxonomy .................................... 165
C. Case Study Data ............................................................ 173
Bibliography 179
LIST OF FIGURES

2.1. Sytematic Literature Review ................................................. 19
3.1. Geographical Distribution .................................................. 47
3.2. Yearly frequency ............................................................. 47
3.3. Nature of the Work ......................................................... 48
3.4. Classification of previous Works ......................................... 50
3.5. Time Evolution ............................................................... 51
3.6. Software Reliability frequently associated Concepts ............... 52
3.7. Academic Works Distribution .............................................. 52
3.8. Industrial Works Distribution ............................................. 53
3.9. SQuaRE Quality Model ..................................................... 56
3.10. Research Keywords ......................................................... 59
3.11. Research Topics ............................................................ 63
4.1. Model Proposal Layout ....................................................... 79
4.2. The Model in its context ..................................................... 81
5.1. The Failure Mechanism ....................................................... 100
5.2. Reference Defects Taxonomy ............................................... 104
6.1. Data Gathering Process ..................................................... 118
6.2. Software Reliability Structural Model .................................. 121
6.3. Stakeholders View on Reliability ....................................... 123
6.4. Expectation vs Assessment ............................................... 125
6.5. Functional Usage Model .................................................... 136
6.6. Summary of Analysis Computations .................................... 139
6.7. Defect Classes Impact on Reliability ........................................ 142
6.8. Normalized Impact on Reliability ........................................ 143
<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Selected Reviews on Software Reliability</td>
<td>39</td>
</tr>
<tr>
<td>3.2</td>
<td>SMS Research Questions</td>
<td>40</td>
</tr>
<tr>
<td>3.3</td>
<td>SMS Inclusion Criteria</td>
<td>41</td>
</tr>
<tr>
<td>3.4</td>
<td>SMS Exclusion Criteria</td>
<td>41</td>
</tr>
<tr>
<td>3.5</td>
<td>SMS Search Strings</td>
<td>42</td>
</tr>
<tr>
<td>3.6</td>
<td>SMS Selection Summary</td>
<td>43</td>
</tr>
<tr>
<td>3.7</td>
<td>Nature of the Work</td>
<td>48</td>
</tr>
<tr>
<td>3.8</td>
<td>Classification of previous Works</td>
<td>50</td>
</tr>
<tr>
<td>3.9</td>
<td>SQuaRE Reliability Characteristics definitions</td>
<td>57</td>
</tr>
<tr>
<td>3.10</td>
<td>SQuaRE Reliability Views</td>
<td>58</td>
</tr>
<tr>
<td>3.11</td>
<td>SLR Research Questions</td>
<td>58</td>
</tr>
<tr>
<td>3.12</td>
<td>SLR Inclusion Criteria</td>
<td>60</td>
</tr>
<tr>
<td>3.13</td>
<td>SLR Exclusion Criteria</td>
<td>60</td>
</tr>
<tr>
<td>3.14</td>
<td>The Selection Process</td>
<td>60</td>
</tr>
<tr>
<td>3.15</td>
<td>The Selected Works</td>
<td>62</td>
</tr>
<tr>
<td>3.16</td>
<td>Research type and objective</td>
<td>63</td>
</tr>
<tr>
<td>3.17</td>
<td>Answer to the Research Questions</td>
<td>67</td>
</tr>
<tr>
<td>5.1</td>
<td>Fuzzy Assessment</td>
<td>96</td>
</tr>
<tr>
<td>5.2</td>
<td>Random Index</td>
<td>97</td>
</tr>
<tr>
<td>5.3</td>
<td>Search On Software Defects Taxonomies</td>
<td>101</td>
</tr>
<tr>
<td>5.4</td>
<td>Selected Previous Software Defects Taxonomies</td>
<td>102</td>
</tr>
<tr>
<td>5.5</td>
<td>Case Study Kinds</td>
<td>110</td>
</tr>
<tr>
<td>5.6</td>
<td>The Incremental Case Study</td>
<td>111</td>
</tr>
</tbody>
</table>
6.1. Research Questions and motivation ................................................. 114
6.2. Pairwise comparison ................................................................. 122
6.3. Computed Priorities ................................................................. 123
6.4. Reliability Computation ............................................................ 124
6.5. Fuzzy Assessment ................................................................. 127
6.6. Assessment example L1 ............................................................. 128
6.7. Assessment example L2 ............................................................. 128
6.8. Random Index ................................................................. 129
6.9. First Level Assessment ............................................................ 130
6.10. Second Level Assessment. Left branch ........................................ 130
6.11. Second Level Assessment. Right branch ....................................... 130
6.12. The Global Weight Vector ....................................................... 131
6.13. Computed Relative Importances ............................................... 131
6.15. FAHP Consensus on User Perception ........................................ 132
6.16. De-Fuzzified Consensus ......................................................... 132
6.17. Users Subjective Assessment at $T_0$ ........................................... 134
6.18. Users Subjective Assessment at $T$ ............................................. 134
6.19. FAHP Consensus on Reliability evolution .................................... 135
6.20. Frequency Modifiers ............................................................... 137
6.21. Collected Operational Data ...................................................... 138
6.22. Defect Global Assessment ....................................................... 138
6.23. FAHP Consensus on Reliability evolution .................................... 140
6.24. Reliability Assessment from Defect Data .................................... 140
6.25. Reliability assessment at C.S. endpoint ...................................... 141
Part I

Theoretical Basis and Proposal
CHAPTER 1

INTRODUCTION

This chapter presents the research motivation and an overview of the research background, along with its hypothesis and objectives. The sponsored research projects that contribute toward formalizing the ideas behind this research also are presented. Finally, the structure of this work is outlined.

No puede impedirse el viento, pero podemos construir molinos.
Proverbio

1.1. BACKGROUND AND MOTIVATION

This work is concerned with a major topic in Software Engineering that of the Software Product Quality and more specifically on Software Product Reliability which according to Musa [90] is probably the most important of the characteristics inherent in the concept Software Quality.

It should not currently be necessary to discuss the paramount importance that Software Reliability represents [76] in many sectors of industry and society. Software products have become extremely pervasive, and with them our dependence on software driven systems. It is generally accepted that Reliability is a key factor in Software Quality since it quantifies failures and misbehaviour. Also, on the economic point of view, high Reliability is desirable if the total costs of the software product are to be reduced. There are also other very important aspects such as customer dissatisfaction and loss of the manufacturer’s prestige that can be traced to software product reliability issues. From Software Engineering perspective Software Reliability is [77] the crucial factor as regards estimating both software cost and Software Quality.
Quality and so Reliability, is a complex difficult to define and understand [43] concept with many different dimensions. It is characterized by a strong subjective element which needs to be taken into account when modeling it. David Garvin analysed in his now classic work [50] how Quality is perceived from several domains, presenting a view of Quality as an abstract concept that comprises five different perspectives; transcendental, user-based, value-based, manufacturing and product-based perspectives. Garvin [50] proposes the following definition for each of these concepts:

1. The transcendent view of Quality is its innate excellence, absolute and universally recognizable as well as cannot be defined precisely; rather, it is a simple, unanalyzable property that we learn to recognize only through experience.

2. The user-based approach is an idiosyncratic and personal view of quality, and one that is highly subjective. Individual consumers are assumed to have different wants or needs, and those goods that best satisfy their preferences are those that they regard as having the highest quality.

3. The Value-based approach defines quality in terms of costs and prices. According to this view, a quality product is one that provides performance at an acceptable price or conformance at an acceptable cost.

4. Virtually all manufacturing-based definitions identify quality as "conformance to requirements" since being primarily concerned with engineering and manufacturing practice. Excellence is equated with meeting specifications, and with "making it right the first time."

5. Finally, the product-based view, on the contrary considers quality as a precise and measurable variable. According to this view, differences in quality reflect differences in the quantity of some ingredient or attribute possessed by a product. This approach lends a vertical or hierarchical dimension to quality in terms of such desired or possessed attributes. Quality differences could, therefore, be treated as differences in quantity, considerably simplifying the mathematics.

The product-based and the manufacturing process views are traditionally dominant in
Software Quality. Then, ISO 14598-1 [61] defines External quality as the extent to which a product satisfies stated and implied needs when used under specified conditions. Nigel Bevan remarks in [13] that this moves the focus of Quality from the product in isolation to the satisfaction of the needs of particular users in particular situations. However, if different groups of users have different needs, then they may require different characteristics for a product to have quality for their purposes. Assessment of quality thus becomes dependent on the perception of the user. Those concepts meet Garvin [50] user-based perspective and his user perceived quality. Moreover, there is a more fundamental reason for being concerned with user-perceived quality [13]; products can only have quality in relation to their intended purpose.

It is then the user based and the product based perspectives and the relationship among them which is of our interest. Software Product is understood as being the whole set of source code, design documents and user manuals and operation procedures. It is on these entities where we put our interest not in the process to produce them.

This is not putting in cause that developing performable ways in which to build reliable systems is therefore a real need [78], just remarking that knowing how to assess the actual reliability level of any software product is of no less importance. If this is to be achieved then it is necessary to develop models that are able to assess what level of reliability can be delivered by the software systems. Such is the purpose of Software Reliability Modeling (SRM). In an industrial context valuable model have to, in addition to being founded on sound assumptions, capture the phenomena complexity and be demonstrably accurate, being practical and cost effective since the main goal of industrial model is informing the decision making process in a real-world context in a way that is clearly valuable and profitable for the organization.

Only those models are to be effectively applied in day to day industrial practice.

The work of the Standardization Organizations, bringing together the efforts of hundreds of volunteers representing varied viewpoints and interests, are good examples of the necessary efforts to afford this challenge and achieve such models. In industrial context International Standards play a central role in particular the demonstration of compliance with Quality Standards which is commonly recognized as a gauge of higher performance levels as well
generates better positioning in the market by means of increased customer confidence. Companies and organizations increasingly require their providers to comply with International Quality Standards.

Despite the above and the high research activity on Software Reliability there would appear to be very little activity in SRM based on Software Quality Standards, which is surprising given the aforementioned relevance of such International Standards in industry. Work needs to be done to afford this situation in order to enable industry to effectively apply Quality Models to industrial practice. This work is therefore the result of having searched for an answer to the question of how to apply Software Product Reliability Modeling to industrial environments using representative International Standards as a basis, in other words, to investigate the suitability of **Standard Based Software Reliability Modeling (SB-SRM)** as an engineering methodology in real-world contexts.

To do so we have selected the ISO/IEC 25000 "Software Product Quality Requirements and Evaluation" (SQuaRE) [59] series of standards over which to build our study. Notwithstanding other standards will also be considered for suitability and completeness. The rationale behind this selection is that the International Standards, and SQuaRE in particular, tackle the well-known lack of consensus and the variety of views on what Software Quality is, also SQuaRE is the most recent release on this field and then, arguably, offers the more mature proposal in the framework of Standard Based Software Product Quality.

In the SQuaRE proposal [59, 60] the Quality of a system is understood as the degree to which the system satisfies the stated and implied needs of its various stakeholders. It is, thus, necessary to consider Quality from different stakeholder perspectives. Among the documents in the ISO/IEC standard the 25010 Quality Model [60] defines a Product Quality Model composed of eight characteristics which are further subdivided into sub-characteristics, considering in that way a hierarchy of elements. Such hierarchy is a way of structuring the constituents elements for a higher level part of the whole. It is then, an structural model.

Bearing the aforementioned issues in mind, **the aim of this thesis is to support the industrial application of Reliability Modelling as a common practice in software production and maintenance cycles.** To achieve this, we propose the application of structural models strongly
oriented to the user dimension and lightweight application frameworks.

1.2. HYPOTHESIS AND OBJECTIVES

The work conducted in this research project was guided by the following principal hypothesis:

It is possible to assess Software Product Reliability in Industrial Environments in a sound, efficient and effective manner by following an International Standard Based Model.

The general objective of this research is therefore:

To develop a Standard-Based Model for Software Product Reliability description and analysis as well as defining an application framework focused on its Industrial Applicability

The general objective is achieved by the following specific objectives:

- O1. To perform a literature review of the software product reliability in order to produce a State-of-Art and identify trends in the research, proposed approaches and already identified issues.
  - O1.1 To carry out a Systematic Mapping Study (SMS) designed with the aim of identifying and categorizing a broad set of primary studies covering the work currently being considered by researchers as regards the various aspects of the reliability modelling of software driven systems, with a particular focus on Software Product Reliability.
  - O1.2 To carry out a Systematic Literature Review (SLR) with the aim of identifying the current approaches to Software Reliability Modelling related to International Quality Standards and constitutes the reference State of the Art for the present research.

- O2. To identify the impairments for broader application of the Standard-Based Soft-
ware Reliability Modelling (SB-SRM) into real-world industrial environment.

- O3. To define a software product reliability model to overcome such hindrances.

- O4. To develop an efficient framework that can be applied without introducing notable overload and thus promoting the SRM in daily industrial practice.

- O5. Verification and Validation of the Proposal.
  - O5.1 Verify the proposed solution correctness by means of formal peer reviews.
  - O5.2 Validate the proposed Model and Framework by conducting Industrial Case Study aimed to analyse: Its Industrial applicability and its conceptual descriptiveness.

### 1.3. SPONSORED RESEARCH PROJECTS

This research has principally been conducted within the framework of the following research projects:

**SOS project.** Funded by the Consejeria de Educacion, Ciencia y Cultura de la Junta de Comunidades de Castilla La Mancha and the FEDER Fund (European Regional Development Fund) SB PLY/17/180501/0000364).

**BIZDEVOPS-GLOBAL.** Funded by the Spanish Ministry of Science, Innovation and Universities and the FEDER Fund (European Regional Development Fund) RTI2018-098309-B-C31.
1.4. **THESIS STRUCTURE**

This document is structured in two parts; theoretical and empirical and seven chapters and three appendix whose content is shown as follows:

**PART I: Theoretical Basis and Proposal**

**Chapter 1** Presents the background, motivation, research hypothesis and objectives that have guided this research.

**Chapter 2** Presents an overview of the research methods applied during the different parts of this research. The methods include systematic approaches for conducting literature reviews and empirical studies in particular Case Study.

**Chapter 3** Depicts the State-of-the-Art on the topic of Software Product Reliability Modeling as outcome from the conducted SMS and SLR. It also describes how the research fit into Software Quality knowledge areas. Presents the Thematic Synthesis of the results obtained after conducting the reviews and how that leads to identify notable impairments to SB-SRM industrial application.

**Chapter 4** Presents the problem in its context as well as the impairments to be removed in order to achieve our objectives. Then, it presents a solution proposal in the form of an innovative layout to structure the Reliability model. It also presents the Industrial Environment and some relevant considerations to take into account when applying the proposed framework.

**PART II: Application and Empirical Validation.**

**Chapter 5** Presents the Validation Process of the proposed Model and Framework as an Incremental Case Study strategy developed in this thesis to allow its empirical validation. Additionally the tools and practical agreement in relevant concepts for the application of our proposed solution are also presented and discussed in this chapter to conform the necessary operationalization of the model.
Chapter 6 Presents the implementation and the results obtained after conducting the Incremental Case Study that validates; first the feasibility of the approach, then its ability to capture the Stakeholder’s expectations on Software Product Reliability and finally the relationship between the low level attributes and the exhibited behaviour as perceived by the user. This Case Study permits, then, to understand the extent to which the proposed Model and application method is able to overcome the main identified hindrances as well as its industrial applicability.

Chapter 7 Presents the main contributions of this work, the results obtained and outline some future research.

Appendices

Appendix A: Literature Review Search.
Appendix B: Software Defects Taxonomy.
Appendix C: Case Study Data.
CHAPTER 2

RESEARCH METHODOLOGIES

This chapter presents an overview of the research methods used to fulfil the objectives of this thesis. These methods can be categorized as Evidence Based Software Engineering and Empirical Software Engineering methods. Within the first category, Systematic Mapping Studies, Systematic Literature Review and Thematic Synthesis are presented here in the aim to be applied to identify and analyse studies about Software Product Reliability Modelling and its relations with the International Software Quality Standards with a particular interest in the ISO/IEC 25000 family. The methods in the second category are aimed to the empirical validation of the proposed solution. Among them Case Studies oriented to assess the feasibility and applicability of the proposed approach reveals as the better suited approach.

The method of science is tried and true. It is not perfect it’s just the best we have. And to abandon it, with its sceptical protocols, is the pathway to a dark age.
Carl Sagan.

2.1. INTRODUCTION

Reviewing literature in order to identify publications related to a specific topic is a common research task that is very often accomplished without methodological support in a non-systematic manner. This kind of informal review does not provide the means to avoid bias during the selection of the publications that will be analysed [39, 67]. From a scientific perspective, the literature review should be complete, systematic and replicable in order to find relevant studies related to the phenomenon being studied [67]. It is also important to note that finding the relevant papers in the massive amount of work published is a major problem. It is therefore important to have mechanisms that can be used to summarize and provide an overview of an area or topic of interest [67] in a fair and unbiased manner.
Evidence Based Software Engineering (EBSE) provides support to successfully achieve this task by proposing [22, 39, 96] a variety of analysis and research methods, the more relevant of which are the Systematic Mapping Study (SMS) and the Systematic Literature Review (SLR).

On the other hand this research is about engineering, which is a practical discipline and in addition, among our main goals there is the validation into a real-world setup which naturally leads us to the application of empirical methods. The standard of empirical research in software engineering includes [122] Case Studies, Surveys, and formal Experiments, whether observed in the field or in a laboratory.

We choose to develop a Case Study because according to [127] a case study design should be considered when: the focus of the study is to answer "how" and "why" questions; we cannot manipulate the behaviour of those involved in the study; and want to cover contextual conditions because believe they are relevant to the phenomenon under study. Case Studies are also recommended when the boundaries are not clear between the phenomenon and context. Our research context fits in all those recommendations thus Case Study is the best suited method to validate our SB-SRM proposal as we will develop on next section 2.3 once EBSE has been presented along section 2.2.
2.2. EVIDENCE BASED SOFTWARE ENGINEERING

Originated from medical research, Evidence-Based Software Engineering [39] has made a major contribution to the methodological evolution of the software engineering area. As developed in [2, 22, 96], Evidence-Based signifies the intention of replacing opinion with a scientific epistemology for the creation of knowledge. This paradigm arose in medical research, among other reasons, because medical practitioners were overwhelmed by the large number of scientific studies and the difficulty judging the evidence’s quality and assessing what the evidence means in terms of their specific circumstances. Although evidence-based medicine (EBM) has its critics it is generally regarded as successful and the evidence-based paradigm is employed widely in clinical medicine and other disciplines, such as education or the social sciences.

Within Software Engineering it is difficult to conduct randomized controlled trials or to undertake double blinding as it is the rule in traditional sciences. In addition, human expertise and the human subject all affect the outcome of experiments. That has also contributed to the application of this evidence-based paradigm to the research in Software Engineering however, it is important to recognize that software engineering research is not the same as medical research. In that sense Budgen et al. [22] shown that software engineering is much more similar to the Social Sciences than it is to medicine. Because of this reason, EBSE had incorporated ideas and information from text books authored by researchers from the area of social sciences.

Evidence based research is [67, 68] the process of systematically reviewing, assessing and summarizing available research findings. The evidence, in our context a model, a technique or a case study, is divulged in a publication paper. Those papers which contribute to a systematic review are called primary studies, while systematic reviews and, in general, any study based on the analysis of previous research is a secondary study. Evidence-Based Software Engineering research pursues the synthesis of the best quality scientific studies on a specific topic and aims thus to present a fair evaluation of a research topic by using a trustworthy, rigorous, and auditable methodology. Attributes which makes the core of any scientific method.

The core concept underpinning this is one of performing a secondary study that systemat-
ically finds, assesses and aggregates the results from a set of primary studies, in order to assemble the best available evidence for answering a research question in as unbiased a manner as possible. The well-defined methodology makes it less likely that the results of the literature review are biased, although it does not protect against publication bias in the primary studies. Within software engineering research we consider two different literature reviews; Systematic Mapping Study (SMS) and the Systematic Literature Review (SLR) and a main evidence aggregation process, the Thematic Synthesis.

Systematic Mapping Studies, also known as a scope study is a type of secondary study that can be performed to analyse a research topic [2, 22, 34, 96] in order to provide an overview of a research area and allow us to identify the quantity and type of research and results that are available. SMS is nowadays a standard research methodology in Software Engineering with many publications already available. The main purpose of an SMS is [67] to provide an overview of a topic area and to identify areas for further empirical research. Systematic mapping is a way of taking stock of the research available in any given area, so that we can decide how to develop it. SMS systematically and transparently describes the nature, coverage and quality of research in a broad topic area, identifies gaps in the research base, thus the map is a resource for further analysis. An SMS is, meanwhile, described as being a study that provides a structure of the type of research reports and results that have been published by categorizing them [96]. The Systematic Literature Review is another form of secondary study that uses a well-defined methodology to “identify, evaluate and interpret all available evidence relevant to a particular research question, topic area, or phenomenon of interest” [67]. The goal of an SLR is to integrate existing evidence in order to provide an answer to a previously defined research question. SLRs can be carried out to synthesize existing evidence regarding the benefits or limitations of a given technology, to characterize research addressed in a specific area, to provide a background for new research activities and to evaluate the extent to which theoretical hypothesis are supported by empirical studies [67].

The main differences between an SMS and an SLR are discussed by [66, 96] and are thoroughly discussed in [66], in which we can read that (despite slight methodological differences) both techniques share the same basic methodology, although they have different goals and scope. While an SLR attempts to answer a very specific question related to competing technologies,
an SMS attempts to identify and classify the reported research related to a broad topic [22]. The principal goal of an SMS is the overview and classification of literature, while that of an SLR is to extract and aggregate the best information from the literature available. The scope of the studies is also different: mapping studies usually relate to a broad topic area whereas an SLR usually relates to a much more detailed subject, and there are consequently differences in the nature of the research questions. A standard systematic literature review is driven by a very specific research question while a mapping study, since it reviews a broader topic, will contain several rather generic research questions.

A standard systematic literature review makes an attempt to aggregate the primary studies in terms of the research outcomes, whereas a mapping study does not discuss the outcomes of the primary studies and usually aims only to classify the primary research papers selected. In-depth analysis and synthesis of the evidence is not the aim of systematic mapping but of the systematic review and thematic synthesis methods. In addition, an SMS would be performed before an SLR because the outcome of the SMS can suggest further investigation when knowledge gaps are detected [67] or could show clusters of studies that could be inputs for the SLR method [96].

Thematic Synthesis; Research Synthesis has to do with the accurate combination of study outcomes in this terms synthesis is, therefore, at the core of the scientific enterprise in EBSE. Research synthesis is a collective term for a family of methods for summarizing, integrating, combining, and comparing the findings of different studies on a topic or research question. Research synthesis is, thus, a way of making sense of what a collection of studies is saying [92]. We can consider the synthesis from three main perspectives; as the combination of thesis and antithesis into a higher stage of truth, as the combination of parts or elements so as to form a whole, and finally we can consider that synthesis is the combining of often diverse conceptions into a coherent whole. Two last definitions are those more in-line with which Research Synthesis means in our context. [92] further distinguished between integrative and interpretive syntheses. Integrative syntheses are concerned with combining or summarizing data for the purpose of creating generalizations. Interpretive syntheses, on the other hand, achieve synthesis through subsuming the concepts identified in the primary studies into a higher-order theoretical structure. While most forms of synthesis can be characterized as being either primarily interpretive or primarily integrative, every integrative synthesis will
include elements of interpretation, and every interpretive synthesis will include elements of integration. Thematic Synthesis refers to a particular method or toolkit that is often used to analyze data in qualitative research. It is characterized by its flexibility and ability to cope with large evidence base and evidence types. Consisting on the identification and analysis of major and recurrent patterns within data and themes in Literature, summary of findings of primary studies under these thematic headings and characterized by flexible procedures and being largely descriptive.

The method used to conduct an SMS is therefore described in Section 2.2.1., while Systematic Literature Review methods are shown in Sections 2.2.2 and Thematic Synthesis is presented in 2.2.3.

2.2.1. Systematic Mapping Studies

Since an SMS can be used to support further research activities, it should be of high quality. The citation and classification of all references, the use of a well-defined and robust taxonomy and the application of a stringent search process are expected [66]. The use of specific paper taxonomy [120] and visualization techniques with which to aggregate data are also suggested [96]. An SMS that meets quality criteria can be used to: establish a baseline with which to analyze trends over time, provide a rationale for further empirical research, identify relevant literature in order to elaborate a related work section of other studies, validate the search for results in the SLR itself, and support educational ends [66]. However, if an SMS does not meet the quality criteria, it can be used as starting point for a more detailed SMS.

As we previously said, the objective of systematic mapping is therefore to provide an overview of a research area in order to assess the quantity and type of primary studies that exist on a topic of interest [22] with the aim of classifying the available research and identifying sub-topics in which more primary studies are needed. The outcome of a Systematic Mapping Study is a high level map, usually in the form of a set of tables and graphics, containing condensed information about a research area and visualizing the status of the field with regard to the research questions.

According to [67], a Systematic Mapping Study is conducted by: (1) planning, (2) conducting a search and (3) screening primary studies using inclusion and exclusion criteria. A system-
atic mapping also conducts (4) data extraction and analysis through the identification of
categories and the classification of the primary studies in these categories. This eventually
leads to the last step (5), which consists of building a map containing the results.

1. **Systematic Mapping planning.** In this step, the systematic mapping plan that will be
   used as a basis to conduct the systematic mapping is established. The following tasks
   are typically carried out at this stage:

   a) Defining the scope by means of the Research Questions. These questions need to
      be designed with regard to the objective that the systematic mapping is intended
      to attain.

   b) Search strategy: Selection of sources. We establish which search sources (state-
      of-the-art academic and professional publication databases) will be used to find
      the primary studies. Availability, accessibility and quality criteria need to be
      taken into account at this moment.

   c) Selection criteria. One element that is of paramount importance during the
      systematic mapping planning is the definition of the Inclusion Criteria (IC) and
      Exclusion Criteria (EC). These criteria make it possible to include primary studies
      that are relevant to answer the research questions and exclude studies that do not
      answer them. These criteria must be straightforward to apply and not require
      any interpretation to do so, in order to mitigate the cultural bias of each evaluator
      during the study selection step.

2. **Conducting the search.** This implies searching for relevant papers. In this step, the
   search for primary studies is conducted according to a previously established plan.
   This search is conducted by looking for all the primary studies that match the search
   string in the search sources. This can be carried out automatically if these sources
   have an efficient search engine. This is done by establishing the keywords and the
   search strings.

3. **Selection of the primary studies.** In this step, the selection criteria (i.e., inclusion and
   exclusion criteria) are applied in order to select the relevant primary studies.
4. **Analysis and classification.** The reviewers read the title and abstract in the search for terms and concepts that reflect the contribution of the paper. While doing so, the reviewer also identifies the context of the research. The studies can be grouped into categories that cluster those which are most closely related.

5. **Map building.** Once the classification scheme is in place, the relevant articles are sorted into the scheme, i.e., the actual data extraction takes place. The classification scheme evolves simultaneously to the data extraction through the addition of new categories or the merging and splitting of existing categories.

### 2.2.2. Systematic Literature review

A Systematic Literature Review is a process whose intention is to identify, extract and aggregate the best information from the available literature which, with the aim of mitigating bias, uses replicable methods to identify relevant studies and then to analyze those studies. When an SLR is conducted, the researcher should be aware of publication bias since it could prevent strong conclusions about the topic being studied from being drawn [67]. Publication bias refers to the problem that positive results are more likely to be published than negative results.

An SLR requires more effort than a traditional literature review, but the benefits of an auditable procedure and the strict selection of primary studies support stronger conclusions. Compared with traditional literature reviews, an SLR has the following elements that must be explicitly addressed in the SLR methodology [67]:

1. A review protocol is developed. It describes both a specific research question and the methods to be used to identify, analyse, and summarize evidence.

2. A search strategy is defined whose goal is to identify all potentially relevant literature.

3. The search strategy is documented in order to allow readers to assess the rigor, completeness and replication of the process.

4. The selection of primary studies is based on explicit inclusion/exclusion criteria.
5. Artefacts are developed and used to gather data from selected empirical papers. The artefacts can be either checklists with which to assess study quality or templates with which to extract data.

This research methodology, following the standard guidelines for SLR as specified in [18, 39, 66, 67, 112] and applied by previous SLRs in the Software Engineering area [21, 25, 98] from which valuable insight has also been obtained, is outlined in Figure 2.1. The main activities of the SLR are grouped in four phases: planning the review, conducting the review, analysing the data, and reporting the review [67]. Each phase consists of several activities, as is depicted in figure 2.1.

**Figure 2.1: Systematic Literature Review**

**Planning the Review** As already said, an SLR can be carried out for two principal reasons: to draw more general conclusions about a phenomenon derived from primary empirical papers, or to establish a background for further research activities. These can be considered as a starting point as regards defining the need for a review. To address the need for a review, researchers should check for the existence of a published SLR about the research topic [67]. In addition, the general need should be described in an appropriate context, determined by the phenomenon under study, which would support the justification needed to carry out the SLR. The review of existing SLRs about the research topic can also help in the definition of a specific research question.
The most important part of an SLR is the specification of the research question because the entire review process depends on it. The research question supports activities related to the search strategy, the data extraction procedures and the data synthesis [67]. Given that an SLR demands more effort than a traditional literature review, it is important to take into account the relevance of the SLR results in terms of the potential importance for target audiences (practitioners or researchers), the likelihood of it influencing current software engineering practices, and the potential support for current practices that are believed to be the best with which to perform certain tasks [67].

After the research question has been specified, a review protocol must be developed. The main goal of the review protocol is to reduce researcher bias as regards identifying, selecting, analysing and synthesizing evidence from empirical primary papers. The components of a protocol are the following: background to the study, research question(s), search strategies, study selection criteria, study selection procedures, study quality assessment checklists and procedure, data extraction strategy, synthesis strategy and procedure, dissemination strategy and project schedule [67].

**Conducting the Review** The purpose of an SLR is to apply an unbiased and rigorous search strategy to find as many primary studies addressing the research question(s) as possible. The search on Literature Repositories is based on defining a search string which can be defined by means of feedback from preliminary searches, keywords related to the research questions presented in journals or conference papers, synonyms, abbreviations and alternative spellings should also be considered when defining the search string. Once the search string has been defined, the search process can be performed in digital libraries, reference lists, journals, grey literature, conference proceedings, etc.

In order to be replicable, the search process must be documented and products generated during the process must be saved. This includes the list of unfiltered search results, along with the rationale used to search in specific digital libraries, journals and conference proceedings. In summary, the reasons why decisions about the procedures applied in the review were made and the criteria used to include/exclude papers should be documented.

The retrieved list of papers must be selected with regard to a set of criteria that is relevant to the investigation. The selection criteria are applied to identify the articles which directly
address the research question. In order to minimize selection bias, the selection criteria must be defined when the review protocol is developed. The selection criteria may evolve when conducting the review and the research could lead to a refinement of the selection criteria. A common practice is that of applying selection criteria to the title and abstract of an empirical paper. If they do not provide sufficient information to enable a decision to be made then the conclusions should also be considered [67]. In addition, it is suggested that the reliability of inclusion decisions be checked using an inter-rater approach or test-retest approach [67].

The quality assessment of candidate papers is a difficult task since there is no agreed definition of study quality [67]. A quality study could be one that minimizes bias and maximizes internal and external validity. Bias is a systematic error that tends to produce results that are inconsistent with the real results of the population. Internal validity is related to the design and conduction of the study in order to prevent systematic errors, while external validity is the extent to which the results obtained can be used in other contexts [67].

**Analysing the Data** Once the selection criteria procedure has been applied to retrieved studies, the researchers obtain a set of primary papers. The next step is that of data extraction to identify relevant results from primary studies. It is suggested that two or more researchers carry out the data extraction independently. This approach allows the determination of inter-researcher consistency. Inconsistencies between researchers must be discussed and resolved [67]. Statistical analyses and the classification of different dimensions can be conducted after extracting data if these methods are previously described in the review protocol.

The data synthesis activities involve comparing and summarizing the results of the primary studies. The methods used to perform data synthesis should be documented in the review protocol. The outcome of the synthesis can be a descriptive summary or a quantitative summary [67]. The quantitative summary can be developed by applying methods such as meta-analysis. The descriptive synthesis can tabulate data, considering the research question in order to highlight similarities and differences between the results reported in primary papers. The synthesis involves attempting to integrate results and conclusions in
those cases in which different researchers may use terms with a certain level of similitude.

**Reporting the Review** In this last phase, a report is written in order to disseminate the SLR results to the target audiences previously identified. Reporting the result in academic journals or conferences is a common practice. In addition, the SLR should be reported as a technical report or in a section of a PhD thesis. It is suggested that the report should include: title, abstract, background, research questions, review methods, excluded/included studies with their respective rationale, discussion, and conclusions, among other sections.

In summary; the first stage comprises both a formulation of the problem and the establishment of a protocol that will drive the review. The objectives and the scope of the review are identified at this stage and are expressed by means of the Research Questions (RQ) from which keywords are derived. A Systematic Literature Review is driven by a very narrow research objective that is formalized by means of a short set of very specific research questions. It is also necessary to plan a search strategy by selecting which search sources will be used to find the primary studies. Inclusion Criteria (IC) and Exclusion Criteria (EC) are formalized in order to make it possible to include only primary studies that are relevant to answer the research question. These criteria must be straightforward to apply and, to mitigate each evaluator’s bias, not require any interpretation.

The second stage is the data collection and evaluation process. This signifies searching for relevant papers that match the search string in each of the search sources selected. This is done by establishing the keywords and the particular search strings for each literature source. Once the search has been completed we can proceed to the Selection of the primary studies by applying the inclusion and exclusion criteria. The reviewers analyze the title and abstract in the search for terms and concepts that reflect the contribution of the paper. Once the selection phase has finished, the resulting works are analyzed to extract the data that is relevant to the research objectives. Finally, the overall process and outcomes are reported. It is important to note that this is, usually, an iterative process involving several reviewers and several review meeting intended to get the better consensus on the selection of the primary sources.
2.2.3. Thematic Synthesis

Still among the EBSE techniques the Thematic Synthesis (TS) is typically applied as the closure of a SLR as we have mentioned previously. The purpose of the findings section in a SR is to present what has been discovered through the process of synthesis. In SE, primary studies are often too heterogeneous to permit a statistical summary and, in particular, for qualitative and mixed methods studies, different methods of research synthesis are required [39].

The traditional view of research synthesis is the integrative, quantitative approach with its emphasis on accumulation of data, and analysis in terms of a meta-analysis. Meta-analysis is a form of additive synthesis that combines the numerical results of primary studies, estimates the descriptive statistics and explains the inconsistencies of effects as well as the discovery of moderators and mediators in bodies of research findings [35]. The purpose of meta-analysis is to aggregate the results of studies to predict future outcomes for situations with analogous conditions.

Contrary to the purely integrative, quantitative method, we find several methods for conducting interpretive syntheses of qualitative and heterogeneous research. Traditionally, the interpretive synthesis has been presented in the form of a narrative, as opposed to statistical, summary of the findings of studies [101]. Its form may vary from the simple recounting and description of findings through to more interpretive and explicitly reflexive accounts that include commentary and higher levels of abstraction [113]. A recent evolution within the synthesis literature involves the application of synthesis methods to combine evidence of both qualitative and quantitative methods to obtain a more complete picture of a phenomenon. Thematic analysis is [4, 17] one such method for identifying, analysing and reporting patterns (themes) within data. However, an important problem with respect to research synthesis is that a thematic analysis has limited interpretative power beyond mere description if it is not used within an existing theoretical framework.

The synthesis methods discussed above is by no means meant to be an exhaustive list; there exists numerous other methods with slightly different terminology. Besides, although combined approaches are methodologically challenging, in that they combine both integrative and interpretive synthesis, they may help move Software Engineering research [39] beyond
the currently competing polarities of quantitative and qualitative empirical research.

At the heart of the findings section in a SR, there is always a narrative of the discoveries made. Sometimes these are a compelling narrative of the topic under investigation; other times just a brief description of tables. In some cases we could recognize some logical structure in the text such as a narrative in a chronological order of the evidence, while in other cases [35] we could not recognize such logic. In addition to the narrative, tables and charts represent the simplest types of graphic presentations. Tables allow for condensing and organizing data about many themes into rows or columns. They provide important structure and sequencing that makes the logic trail easier for readers to follow.

Braun and Clarke [17] consider this synthesis as a “Thematic Analysis” consisting on the identification of major and recurrent themes in literature, summary of findings of primary studies under these thematic headings, characterised by flexible procedures for reviewers and being largely descriptive and data-driven basis to groupings. Rodgers [101] presents it as Narrative synthesis, consisting of selected narrative description and ordering of primary evidence with commentary and interpretation. It is characterised by its flexibility, can cope with large evidence base and diverse evidence types. Can also be used for theory-building but still suffers of a lack of transparency as well as procedures or standards. Maybe dependent on the reviewer prejudices.

2.3. EMPIRICAL SOFTWARE ENGINEERING

On this section we present some fundamental concepts on Scientific Research with a particular focus on the Empirical aspects and how they fit in the specific domain of the Software Engineering.

In scientific research, we often refer to the two broad methods of reasoning as the deductive and inductive approaches. Induction is usually described as moving from the specific to the general, while deduction begins with the general and ends with the specific; arguments based on experience or observation are best expressed inductively, while arguments based on laws, theorems, or other widely accepted principles are best expressed deductively. In other words, the deductive approach is analytic reasoning from theories or established models to
its consequences. Inductive reasoning is essentially the kind of reasoning used if you have gradually built up an understanding of how something works, collecting relevant experience and try to construct principles from it. The inductive approach emphasizes on observation and deriving conclusions through observation. That is, empirical scientific method where inductive reasoning progresses from real world observations to the development of a generality. Such is our aim, applying the observation of the behaviour of a specific software driven system with the objective of gaining general knowledge about it that is an empirical research on Software Engineering.

Empirical studies can be applied in order to increase knowledge at a practical and theoretical level. Improving the understanding of current practices in order to develop software is the basis for changing and improving the way we work [123].

2.3.1. Characteristics of Empirical Research.

Empirical research can be qualitative or quantitative. As a rule, quantitative research is mainly concerned with the degree in which phenomena possess certain properties, states and characters, and the similarities, differences and causal relations that exist within and between these. The advantage of the quantitative approach is that it measures, thus facilitating comparison and statistical aggregation of the data. This gives a broad, generalizable set of findings. Qualitative research on the other hand, is mainly concerned with the properties, the state and the character (i.e., the nature, of phenomena). The word qualitative implies an emphasis on meanings that are rigorously examined, but not measured in terms of quantity, amount or frequency. Qualitative data provide depth and detail through direct quotation and careful description of situations, events, interactions and observed behaviours.

The data collected in an empirical study may be then quantitative or qualitative. Quantitative data rely on numbers and classes while qualitative data appear as words, descriptions, pictures or diagrams. Quantitative data can be analysed using descriptive statistics methods or inferential methods, while qualitative data is analysed by sorting and categorization. However, a combination of qualitative and quantitative data often provides better understanding of the studied phenomenon [111], i.e. what is sometimes called "mixed methods" [100].
Traditional quantitative methods such as randomised controlled trials are the appropriate means of testing the effect of an intervention but a qualitative exploration of observations and behaviour is likely to be needed to find out why something happens or how it do. The two approaches should be regarded as complementary rather than competitive, in that way empirical studies tend mostly to be based on qualitative data, as these provide a richer and deeper description, but less precise than quantitative data.

In order to perform an empirical study it is important to consider the different, by its purpose, types of research paradigms. According to [122] there are two; Exploratory research and Explanatory research. Another classification [103] considers four types of purposes; Exploratory, Explanatory, descriptive and improving as also Robson [100] does on his proposed classification. It is to note that even if classes are equally named there are some differences on the corresponding definitions. The following definitions are our own consolidation from these three commonly cited sources.

**Exploratory research.** This is focused on studying objects in their real context finding out what is happening. Observation is the means used to obtain evidence, identify new insights and to generate hypothesis for new research. It requires a flexible research design which is typical of qualitative research.

**Explanatory research.** This is used when seeking for an explanation of a situation or a problem. This aims to explain why a phenomenon occurs mostly but not necessary in the form of a causal relationship. It requires a rigid research design, also known as quantitative research. A typical case of this type of design is that of controlled experiments, in which one or several factors can be manipulated in order to test their effect on the subject under study. The data obtained during the execution of these methods are numbers which can be analysed using traditional statistical methods.

**Descriptive.** This has the objective of portraying and understanding and characterizing a studied phenomenon.

**Improving.** This has the objective of intervening in a studied phenomenon in order to improve a particular aspect of it.
According to the main literature in the topic [100, 122, 123, 127] there are four major empirical research methodologies depending on the purpose of the evaluation and the conditions for the empirical investigation, such as control of the research design or the realism of the study: survey, case study, experiment and Action Research.

**Survey**, which is the "collection of standardized information from a specific population, or some sample from one, usually, but not necessarily by means of a questionnaire or interview" [100].

**Experiment**, or controlled experiment, is characterized by "measuring the effects of manipulating one variable on another variable" [100] and that "subjects are assigned to treatments by random." [122]. Quasi-experiments are similar to controlled experiments, except that subjects are not randomly assigned to treatments. Quasi-experiments conducted in an industry setting may have many characteristics in common with case studies.

**Action research**, with its purpose to "influence or change some aspect of whatever is the focus of the research" [100], is closely related to case study. More strictly, a case study is purely observational while action research is focused on and involved in the change process. In software process improvement and technology transfer studies [51], the research method should be characterized as action research. However, when studying the effects of a change, e.g. in pre- and post-event studies, we classify the methodology as case study.

**Case study** is [12, 100, 127] an empirical method aimed at investigating contemporary phenomena in their context. Robson calls it a research strategy and stresses the use of multiple sources of evidence, Yin denotes it an inquiry and remarks that the boundary between the phenomenon and its context may be unclear, while Benbasat et al. make the definitions somewhat more specific, mentioning information gathering from few entities (people, groups, organizations), and the lack of experimental control. Case study methodology was originally used primarily for exploratory purposes, and some researchers still limit case studies for this purpose. However, it is also used for descriptive purposes, if the generality of the situation or phenomenon is of secondary importance. Klein [71] define three types of case study depending on the research perspective, positivist, critical and interpretive.
Case Study strategies fit particularly well in software engineering where it is the characteristic scenario having to analyse how and why a certain phenomenon occurs but without the capability of controlling it, just from an observer position. On the following we’re going thus to develop in detail the Case Study methodology.

2.3.2. The Case Study Methodology

The term “case study” is used in parallel with terms like field study and observational study for a broad range of studies in software engineering which have in common that they study a specific case, in contrast to a sample from a specified population. Instead, research findings are obtained through the analysis in depth of typical or special cases.

**Characteristics of a Case Study**

The case study is an observational study which could study a project or a product to track a specific attribute or to establish a relationship between different attributes. A case is the object of study. It could be a project, team, activity or process, and it contains one or more units of analysis. The phenomenon under study may be based on existing theories. The case and the units of analysis should be selected intentionally. The purpose of the selection may be to study a case that is expected to be typical, critical, revelatory or unique in some respect [12]. In practice, cases are often selected on the basis of availability.

The main advantage of case study research is that it works in real settings, with real projects. That is, no laboratory environment is set up by the researcher, where factors can be controlled. Instead the phenomena are studied in their normal context, allowing the researcher to understand how the phenomena interact with the context. However, this setting is usually complex and non-deterministic.

A case study draws on multiple sources of evidence to investigate one instance of a contemporary software engineering phenomenon within a real-life context, especially when the boundary between phenomenon and context cannot be clearly specified [104]. The major characteristics of case study research are: a flexible type of research to deal with the complexity of real settings, a multiple source of evidence aggregated to draw conclusions, and that it is supported by existing theories in order to add new insights and knowledge to the software engineering field [104]. A major outcome of case study research is the
identification of key factors that may have any effect on the phenomenon being studied. A case study can thus be applied when the researcher’s purposes are exploratory, descriptive or explanatory.

It is important to mention the need of taking into account Ethical Considerations in a Case Study; a case study may uncover sensitive data from organizations that could affect several stakeholders. It is thus important to determine how issues such as confidentiality, anonymity and the reporting of case study results should be addressed. It is better to prevent problems by defining a protocol to follow when ethical issues arise. Key ethical factors include [103]: Informed consent, Subjects and organization explicitly agree to participate in the research. Confidentiality, Opinions collected from individual employees or critical business information must be handled in order to prevent information from leaking and the disclosure of sources. Inducements, some incentives could be provided. However, these need to be analysed and their role as regards threatening the validity of the study should be considered.

The Case Study Activities

Case study methodology is a flexible design strategy where there is a significant amount of iteration over the steps, notwithstanding it is generally admitted that a case study should at least contain the following elements: objective, what to achieve? The case, what is studied? Research questions, what to know? Methods, how to collect data? Selection strategy, where to seek data? There are [123] five major tasks to perform in order to run a case study:

- The Case study design, define objectives and elaborate the protocol that will be used to conduct the case study.

- Preparation for data collection, define procedures, templates and guidelines for data collection.

- Collection of data, execute the plan previously developed in order to gather data.

- Analysis of collected data. With regard to data type collected, apply appropriate methods in order to link evidence from data gathered to conclusions drawn.
Reporting. Consider the target audience to write a report.

Let see in more detail the purpose of those activities and how they could be carried out.

1.- Case Study Design The case study design considers two main aspects of the research: objectives and protocol for the case study. The objective is a statement that describes what is expected when the case study is carried out [103]. For instance, "to evaluate a method for prioritizing requirements" [99]. Goals, aims or purposes can be considered synonyms of the objective [103]. The objective determines the data to be gathered in order to attain it. A case study protocol defines detailed procedures with which to collect and analyze raw data [18, 103] propose an outline for a case study protocol which includes additional sections: analysis, plan validity, study limitations, reporting and schedule. These are described as:

- Analysis methods: define the criteria that will be used to interpret raw data, linking data with research questions and alternative explanations.
- Plan validity: describe procedures in order to minimize the impact of threats to validity.
- Study limitation: specify any threats to validity that may still bias the results.
- Reporting: consider the target audience.
- Schedule: estimate major activities and assign appropriate amount of time.

2.- Preparation for data collection. This step refers to the preparation of templates, management databases, tools or format agreements that could eventually be required to properly gather the required data. Reference documents for the participant as for example refer to the same concept with the same meaning, in the same measure units etc.

3.- Collection of Data: In a case study, conclusions are drawn from several data sources. The data can be gathered from different roles, projects and products. Indeed, the analysis of the differences among different sources contributes to stronger conclusions. Several methods can be used to gather information from stakeholders, such as interviews, observations, archival data and metrics. Questionnaires are also used in case studies [99]. Data collection techniques can be divided into three levels:
**First degree:** Direct methods means that the researcher is in direct contact with the subjects and collect data in real time. This is the case with, for example interviews or Delphi surveys or other related techniques.

**Second degree:** Indirect methods where the researcher directly collects raw data without actually interacting with the subjects during the data collection.

**Third degree:** Independent analysis of work artifacts where already available and sometimes compiled data is used. This is for example the case when documents such as requirements specifications and failure reports from an organization are analyzed or when data from organizational databases are analyzed.

4.- **Data Analysis** With regard to the type of data gathered during a case study, it can be analysed using quantitative or qualitative methods. On the one hand, for quantitative data, typical methods are: analysis of descriptive statistics, correlation analysis, development of predictive models and hypothesis testing [123]. However, a case study will never provide conclusions with statistical significance. On the contrary, different types of qualitative data are linked together to support a strong and relevant conclusion [103]. Hypothesis confirmation is the last phase in qualitative data analysis. Its purpose is to confirm that the hypothesis is true, taking into account additional data [123]. One important technique used in case studies for hypothesis confirmation is that of triangulation, which signifies viewing the studied object from different angles in order to provide a more complete picture of it. Four types of triangulation may be applied [103]:

1. Data (source) triangulation. The evidence is elaborated from more than one data source or gathered from the same data source at different times.

2. Observer triangulation. More than one observer participates in the study and the findings of each one are analyzed and combined. Discrepancies are discussed and resolved.

3. Methodological triangulation. This is based on data gathered from different types of data collection methods.

4. Theory triangulation. The data is analyzed by considering different theories or view-
Since it is easy for confounding factors to influence the conclusions drawn from a case study, the use of a structured approach for qualitative analysis is suggested. This means that the researcher must maintain records of the decisions made, instruments used, links between data, codes, memos, and other documents used in the case study. In order to gather a formalized body of knowledge, the unit of evidence must be coded and iteratively analyzed to generalize a theory [123]. Moreover, sufficient information from each step of the study and every decision made by the researcher must be presented in order to maintain the chain of evidence [103].

5.- Reporting a Case Study The case study report communicates the findings of the study, but it is also the main source used to assess the quality of the study. A large volume of data can be generated during the case study. Some data, however, could be sensitive information. In this case, the use of this kind of data in a report relies on procedures described in the consent agreement. In addition, the target audience determines how to present the report [123]. Reports prepared for peers may contain the following sections: title, authorship, abstract, introduction (problem statement and research objectives, and context), related work (earlier studies, theory) case study design (case and subject selection, data collection procedures, analysis procedure, validity procedure), results (cases and subject description, covering execution, analysis and interpretation issues, evaluation of validity), conclusions and future work (summary of conclusions, relation to existing evidence, impact/implications, limitations, future work) [103]. Finally, the conclusions must be reported and a discussion about the implications of the results should be included.

The case study methodology provides a greater understanding of a studied phenomenon. However, several threats should be controlled in order to gather acceptable knowledge. Special attention should be paid to validity threats, since a case study can be biased by the researchers’ subjective point of view.

2.3.3. Empirical Research Validity

The validity of a study denotes the trustworthiness of the results, to what extent the results are true and not biased by the researchers’ subjective point of view. It is, of course, too late
to consider the validity during the analysis of even after. The validity must be addressed
during all previous phases of the case study. However, the validity is usually discussed
after the case analysis or as the last stage of it, since it cannot be finally evaluated until this
analysis phase.

The validity can be improved by practices such as developing and maintaining a detailed
 case study protocol, using triangulation techniques, obtaining feedback about the protocol
from peer researchers, reviewing collected data and analyzing the results with case subjects,
spending sufficient time with the case, and addressing theories that contradict the case study
findings [103]. Four main aspects must be controlled during case study execution in order
to obtain trustworthy results

**Construct validity:** This aspect of validity reflect to what extent the operational measures
that are studied really represent what the researcher have in mind and what is investigated
according to the research questions. If, for example, the constructs discussed in the interview
questions are not interpreted in the same way by the researcher and the interviewed persons,
there is a threat to the construct validity.

**Internal validity:** This aspect of validity is of concern when causal relations are examined.
When the researcher is investigating whether one factor affects an investigated factor there
is a risk that the investigated factor is also affected by a third factor. If the researcher is not
aware of the third factor and/or does not know to what extent it affects the investigated
factor, there is a threat to the internal validity.

**External validity:** This aspect of validity is concerned with to what extent it is possible to
generalize the findings, and to what extent the findings are of interest to other people outside
the investigated case. During analysis of external validity, the researcher tries to analyze
to what extent the findings are of relevance for other cases. There is no population from
which a statistically representative sample has been drawn. However, for case studies, the
intention is to enable analytical generalization where the results are extended to cases which
have common characteristics and hence for which the findings are relevant, i.e. defining a
theory.

In the case of external validity it is relevant to consider appropriate contextual factors of
the case study since these can affect the generality and utility of the conclusions. Some factors are [68]: the target industry in which products are used, the nature of the software development organization, the skills and experience of software engineers, the tools used in the software process, the software process deployed, among others.

**Reliability:** This aspect is concerned with to what extent the data and the analysis are dependent on the specific researchers. Hypothetically, if another researcher later on conducted the same study, the result should be the same. Threats to this aspect of validity is, for example, if it is not clear how to code collected data or if questionnaires or interview questions are unclear.

It is, as described above, important to consider the validity of the case study from the beginning. Examples of ways to improve validity are triangulation, developing and maintaining a detailed case study protocol, having designs, protocols, etc. reviewed by peer researchers, having collected data and obtained results reviewed by case subjects, spending sufficient time with the case, and giving sufficient concern to analysis of "negative cases", i.e. looking for theories that contradict your findings.

### 2.4. RESEARCH METHODS APPLIED IN THIS THESIS

The conceptual framework of the Research methodology considered the following elements: Research group, research object, stakeholders and external participants. Also techniques and methods are considered depending on which of the two parts of this research are applicable.

**Research Group.**

Of three Software Quality Researchers from Alarcos research group at Universidad de Castilla La Mancha, being the main researcher the author of this Thesis.

**Research Object.**

The problem considered in this thesis is the Analysis of Software Product Reliability from International Quality Standards perspective with a special focus in its Industrial application features.
Stakeholders of the research.

Those are the organizations which may benefit from the outcomes of this research. As primary stakeholder we have to consider industrial companies which production process is highly dependent on Software Product and are interested on improving such product quality. On a more general scope any organization in the need to assess the Reliability of its Software Products or need to prove adherence to International Quality Standards

External Participants.

A selected group of Software Engineering practitioners acted as Domain Experts during the data gathering process of the Empirical Part.

The research presented in this thesis is to be developed along two different phases; the first one of theoretical nature and the second one of empirical research and practical application. On the following we detail the specific methods and tools applied in each part.

Part I - Evidence Based Software Engineering

In the main aim of constitute a scientific valid State of The Art.

- Source of data: Available Literature Repositories

- Analysis methods: Formal SMS and SLR


Part II – Empirical Software Engineering

This second part has aim of proving the Industrial applicability of our Standard-Based Software Reliability Model, as well as to develop a practicable application method for it.

- Source of data.

  - Available Software Project historical records.

  - Available Software Product on-execution logs.
• Software Product experts and managers.

- Analysis Methods

  • Case Study Methodology

  • Fuzzy Analytical Hierarchy Process techniques

- Means. No particular tool was required.
The objective of this chapter is to establish a formal State-of-the-art on the topic of Software Product Reliability Modelling, confirm or refute commonplaces and assumptions and potential misconceptions, define the context and scope of the research and deeply analyse the scientific basis of our research hypothesis. To achieve that we first develop a Systematic Mapping Study which will provide the general context and will unveil potential gaps in the available evidence as well as will provide relevant inputs to the definition and implementation of a Systematic Literature Review focused on our main research question; how to apply Software Product Reliability Modeling to industrial environments using representative International Standards as a basis. Both research are published in [41] and [43].

Science is supposed to be cumulative, not almost endless duplication of the same kind of things.
Richard Hamming 1968 Turing Award Lecture

3.1. INTRODUCTION

This chapter presents the results of both a Systematic Mapping Study (SMS) and a Systematic Literature Review (SLR) on Software Product Reliability modelling (SRM). On the one hand, the SMS is focused on studies addressing product reliability modelling regardless of approaches or application environment. On the other hand, the literature review analyse the application of International Software Product Quality Standards to Software Reliability Modelling. The Quality characteristics used in these reviews are taken, mainly but not exclusively, from the [60] ISO/IEC 25010 software product quality model. The objective is to provide a synthesis of the current support as regards of Modelling Reliability for Product Quality management.
This work kick-off was done in 2014 with the design of a comprehensive Literature Review in two parts; first a SMS then a SLR. The SMS covered a period of ten years from 2003 to 2014. The SLR covered a wider time span, from 1991 to 2014. We choose start at 1991 because corresponds to the first release of ISO/IEC 9126 [62], which is the precedent of our reference ISO/IEC SQuaRE. This specific time span is a trade-off among feasibility and representativeness.

Software Reliability evaluation is of application in fault prediction purposes, determining the optimal time at which to stop testing and release software as well as to providing data with which to make trade-offs between test time, reliability, cost, and performance. Software Reliability models are intended to capture the software properties and characteristics in a useful manner in order to support the aforementioned objectives. The structuring and classification of Software Reliability models is a considerable problem owing to the overwhelming number of models that have been proposed. Attempts have been made to deal with this complex situation, and a variety of classifications have been proposed, but to the best of our knowledge none has been developed using a systematic approach.

Software Reliability Modelling can be divided [65, 77] into two main categories: prediction and assessment models. Both kinds of models are traditionally based on recording fail data and analysing them by means of statistical inference. These statistic models, which are better known as Software Reliability Growth Models (SRGMs), are the most conventional means of reliability analysis. They are used to describe the behaviour as regards an application’s failure during its testing and operational phase, and are black-box based, that is, the software system is considered as a whole, and only its interactions with the outside world are modelled without considering its internal structure. As an alternative to this conventional black-box analysis, architecture-based software reliability analysis considers the software’s internal structure and explicitly relates application reliability to component reliabilities. The objective of this approach is the early assessment of the application’s reliability, which is not possible with traditional models. More recently, approaches such as Bayesian Belief Networks or Test-Based methods have been mentioned in literature [41] as being relevant categories for SRM.
3.2. SMS ON SOFTWARE RELIABILITY MODELLING.

Having presented in previous Chapter 2 the general steps of a Systematic Mapping Study, we shall now apply them to our work. Our systematic mapping aims to identify, structure and classify those primary studies that propose Software Product Reliability Models or Theories as well as provide an overview of the current approaches used to SRM in the context of academic and industrial initiatives based on Software Product analysis.

Indeed, Software Reliability a highly active research area with hundreds of works being published every year makes it difficult to apprehend. Our applied strategy to deal with this sheer number of published works consisted of two stages. In the first we searched for a set of secondary studies from which to derive a generic framework. This search was systematic but sought only to find a sufficient set of studies for such a generic framework, and not an in-depth synthesis of the evidence. The purpose of this analysis is obtaining a first overview on the topic and valuable guidance to positioning the further formal literature review design. The selected studies are those shown in table 3.1.

<table>
<thead>
<tr>
<th>Title</th>
<th>Ref</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>An analysis of competing software reliability models</td>
<td>[48]</td>
<td>1978</td>
</tr>
<tr>
<td>Software reliability status and perspectives</td>
<td>[27]</td>
<td>1982</td>
</tr>
<tr>
<td>Software reliability growth modeling: models and applications</td>
<td>[106]</td>
<td>1985</td>
</tr>
<tr>
<td>Software reliability models for critical applications</td>
<td>[52]</td>
<td>1991</td>
</tr>
<tr>
<td>Software reliability modeling survey</td>
<td>[40]</td>
<td>1996</td>
</tr>
<tr>
<td>Applying software reliability engineering in the 1990s</td>
<td>[119]</td>
<td>1998</td>
</tr>
<tr>
<td>Methods and problems of software reliability estimation</td>
<td>[65]</td>
<td>2006</td>
</tr>
<tr>
<td>Review of quantitative software reliability methods</td>
<td>[33]</td>
<td>2010</td>
</tr>
</tbody>
</table>

Table 3.1: Selected Reviews on Software Reliability.

Despite being highly influential papers none of the above mention the application of any kind of systematic methodology. Bearing all this in mind, we believed that it was necessary to carry out a Systematic Mapping Study (SMS) in order to obtain, among other interesting data, a more complete classification of Software Reliability Models by means of a recognized systematic methodology. Thus, in the second stage, the formal SMS of primary studies was conducted for a time span that was chosen as a trade-off between feasibility and relevance based on the outcomes of the first stage.

The mapping study process suggests that the papers should be screened by at least two
researchers in order to avoid biased evaluations. This systematic mapping was conducted on February 2014 to cover papers published between 2003 and 2014 and involved three people: two software engineering researchers and a PhD candidate. The systematic mapping was conducted using the five steps process presented in Chapter 2, and was completed with the appropriate assessment of threats to validity.

### 3.2.1. Data gathering for the SMS.

#### Step 1: Systematic Mapping Planning.

**Research Questions:** As explained previously, we are interested in general high level questions such as trends in research activity, discovering the concepts that are most frequently associated with our topic of interest, the sort of studies that are available, and so on. These interests need to be explicitly expressed in the form of answerable questions. Since our intention is to identify the most important concepts in Software Product Reliability Models or Theories, the research questions shown in table 3.2 have therefore been defined. For example, in order to explore the research trends, emerging or abandoned approaches or the evolution of the research activity we could ask about the research activity during a particular time span. The following research questions are therefore proposed: How many, who carried them out, and what kinds of SRM activities have there been in the past 10 years? What SRM research topics have been addressed over the past 10 years? Will this provide us with an overview of the main fields of interest in addition to side topics in SRM research? Which SRM models are in use? These questions cover the most important designs and methods along with gaps and underrepresented approaches.

<table>
<thead>
<tr>
<th>RQ</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>How many, who carried them out, and what kinds of SRM activities have there been in the past 10 years</td>
</tr>
<tr>
<td>2</td>
<td>What SRM research topics have been addressed over the past 10 years?</td>
</tr>
<tr>
<td>3</td>
<td>Which SRM models are in use?</td>
</tr>
<tr>
<td>4</td>
<td>Which terms are most often associated with Software Reliability Modelling?</td>
</tr>
</tbody>
</table>

*Table 3.2: SMS Research Questions.*

Regarding this final research question; which terms are most often associated with Software Reliability Modelling? With this, our intention is to identify the most important concepts in the topic. This question is also directly related to the methodology of this mapping study since "key-wording" is intended to lead to the taxonomy by clustering the meaningful terms
belonging to the same category.

Selection of sources: Three of the largest and most complete scientific databases were chosen as the sources of primary studies: IEEE Xplore, ACM Digital Library and Science-Direct, owing to their ease of accessibility and because, as is noted in [18, 39], they are widely recognized as being an efficient means to conduct Systematic Reviews, and thus mappings in the context of Software Engineering. These databases are also commonly used sources when conducting systematic surveys in computing research. However, we considered not only effectiveness, but also the ability to export the results to a well-defined standard format in a straightforward manner.

Establishment of selection criteria: It is at this stage that the inclusion and exclusion criteria are formalized. The inclusion criteria of our systematic mapping are shown in Table 3.3 whereas the exclusion criteria are shown in Table 3.4.

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC1</td>
</tr>
<tr>
<td>IC2</td>
</tr>
<tr>
<td>IC3</td>
</tr>
<tr>
<td>IC4</td>
</tr>
<tr>
<td>IC5</td>
</tr>
</tbody>
</table>

Table 3.3: SMS Inclusion Criteria

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC1</td>
</tr>
<tr>
<td>EC2</td>
</tr>
<tr>
<td>EC3</td>
</tr>
<tr>
<td>EC4</td>
</tr>
<tr>
<td>EC5</td>
</tr>
<tr>
<td>EC6</td>
</tr>
<tr>
<td>EC7</td>
</tr>
</tbody>
</table>

Table 3.4: SMS Exclusion Criteria

Step 2: Conducting the search

The search is conducted using automated search engines. In order to make the search as
unrestricted as possible, the search string is simply "Software Reliability". In practice we also built the particular settings for each search engine (see Table 3.5), since each of them works in a very specific manner. In particular, we attempted to minimize duplications and rejections by setting the appropriate options in each engine e.g. setting the ACM engine so has not to report IEEE published works that would also be reported in the output of its own engine, and also asking only for papers with an abstract or which had been published in the appropriate forums.

<table>
<thead>
<tr>
<th>Database</th>
<th>Search String and settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM digital library</td>
<td>(Abstract:&quot;software reliability&quot;) and (not Publisher:IEEE) and (PublishedAs:journal OR PublishedAs:proceeding OR PublishedAs:transaction) and (FtFlag:yes) and (AbstractFlag:yes) Publication Year: 2003-2014</td>
</tr>
<tr>
<td>Science Direct</td>
<td>pub-date &gt; 2002 and TITLE-ABSTR-KEY (&quot;Software Reliability&quot;) [All Sources (Computer Science, Engineering)]</td>
</tr>
</tbody>
</table>

Table 3.5: SMS Search Strings

As a result of this step, we obtained a total of 972 papers: 114 from ACM digital library, 739 from IEEE Xplore and 119 from Science Direct (see Table 3.6). In all cases bibliographic data, including the abstract, were exported and stored in BibText format for further analysis. The analysis of the primary studies was supported with JabRef, an open source reference manager system that is able to manage, among other things, BibText databases in a very efficient manner.

**Step 3: Selection of the primary studies**

During the screening of the papers for relevant publications regarding our research questions, and bearing the above criteria in mind, we closely examined the title, abstract and keywords of each paper. In the case of papers whose abstracts did not provide sufficient information to make a decision about the content of the paper, it was necessary to read the whole paper to
determine its relevance. The aforementioned inclusion and exclusion criteria were therefore applied at this stage. A paper would be accepted as a primary contributor to the study as long as it met at least one of the Inclusion Criteria. A paper would similarly be rejected when at least one of the Exclusion Criteria was fulfilled.

The selection process comprises five iterations. The first three were carried out by one reviewer, the fourth by two evaluators and the last one was conducted as a joint review by the three researchers. In the first iteration only papers that were clearly not within the scope of the research questions were excluded, and this first review was also very useful as regards developing an understanding of the topic. It was in the next iteration that the first set of excluded and included studies was obtained. It is important to highlight that the studies included were labelled with keywords denoting the reason for their selection. This labelling greatly facilitated any further analysis, particularly the classification work. In this step, undetermined papers (whose abstracts were not sufficiently clear) were labelled as such. A third iteration was carried out by the same reviewer with the aim of remedying eventual mistakes and deciding on those papers that had been considered as undetermined in the previous step. The full text in these papers was reviewed, and the other two researchers then reviewed the results of all the above analyses. The intention of this step was to reduce any bias in the results. Discrepancies regarding the inclusion and labelling of a paper were discussed and agreed on in a final joint review.

A total of 503 papers were eventually selected - 386 from IEEE Xplore, 50 from ACM Digital Library and 67 from ScienceDirect (see Table 3.6).

<table>
<thead>
<tr>
<th>Source</th>
<th>Included</th>
<th>Excluded</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE Xplore</td>
<td>386</td>
<td>353</td>
<td>739</td>
</tr>
<tr>
<td>ACM Dig. Lib.</td>
<td>50</td>
<td>64</td>
<td>114</td>
</tr>
<tr>
<td>ScienceDirect</td>
<td>67</td>
<td>52</td>
<td>119</td>
</tr>
<tr>
<td>Total</td>
<td>503</td>
<td>469</td>
<td>972</td>
</tr>
</tbody>
</table>

*Table 3.6: SMS Selection Summary*

It should be noted that the number of duplicities was very low, as expected with the search settings. Moreover, it is worth mentioning that a notable set of the excluded works was sorted into two categories that can be defined as "ways to attain reliable systems" (which refers to, e.g., techniques such as rejuvenation or how to build fault-tolerant software) and
“software reliability testing” e.g. testing techniques or testing process management) in addition to those that were not papers.

**Step 4: Data extraction. Analysis and classification**

As a result of the analysis of the selected secondary studies it can be concluded that the taxonomy of the SR Models is an open question in which dozens of classification schemes have been proposed in the past. In [48] the authors studied the most widely used reliability models, tracing the historical development of the various models and analysing their advantages and disadvantages. They classified the models as data domain and time domain models. Some years later, the authors of [27] provided a survey of the various software reliability models that have been proposed since the early 1970s. In this work, a classification scheme was created using the lifecycle phase during which the model is applicable as a primary key. The authors additionally discussed these models using the concepts of residual error size and the testing process used. Published in 1985 and focused on SRGM, the work presented in [106] thoroughly describes and then classifies the existing models by using the error detection rate, a concept that the authors introduce in the same work. In the early 1990s Pham and Pham [52] classified the SRM as being Deterministic or Probabilistic depending on the nature of the parameters used. Only two models (Halstead’s and McCabe’s) were classified as being deterministic, while the rest of the almost fifty reviewed were classified in one of the ten subcategories (Software Reliability Growth being one of them) of probabilistic models. In a later work, the history of SRM was reviewed [40]. In this work, the author classified Software Reliability models according to the kind of data the models uses: failures per time unit or time between failures, along with the reuse of the Musa-Okumoto classification scheme based on a selected set of attributes. In [119] a review of the progress in software reliability since 1975 was carried out. An interesting survey of models appears in [65], although this is mainly focused on SRGM. More recently, in [78] three main approaches to Software Reliability Modelling were identified: the known data and time domain classes, and the error seeding and tagging approach. The IEEE Standard 1633-2008 [56] states that “there are the following three general classes of SR prediction models: exponential non-homogeneous Poisson process (NHPP) models, non-exponential NHPP models and Bayesian models”. In [33] the SR models have been sorted into four classes: Software Reliability Growth Methods, Bayesian Belief Network (BBN) methods, Test-Based methods and other methods, a category which considers for example benchmark
practices or metric-based methods.

As mentioned previously the SMS methodology preconizes the classification of the primary studies by applying keyword clustering. This means that the classification categories are obtained using the keyword clustering of the meaningful terms extracted from the title, abstract and author’s keywords when provided by the authors. This process thus relies on what the authors claim about their work. The keyword clustering is carried out in two steps. Once the meaningful terms or keywords have been extracted they are clustered into relevant categories with regard to the research questions, e.g. the terms “prediction” and “assessment” are keywords that are used to classify the works as being related to the prediction or assessment of software Reliability. A classification scheme of this nature can be developed from scratch by considering only the set of included primary studies, or constructed by relying on previously proposed taxonomies for the topic under study. We have chosen the option of driving our classification criteria with a previously proposed taxonomy since we consider it the better approach to deal with the sheer heterogeneity of terms in the context of an SMS. We chose the proposal depicted in [33] because we found that the proposed categories were easy to identify without the need to analyse each paper in depth as occurs in SLRs but also because it is a very recent proposal that has originated from a reliable institution.

The models were therefore first sorted according to [33] into four classes:

1. Software Reliability Growth Models (SRGM).
2. Bayesian Belief Networks (BBN).
3. Test-Based methods (TBM).
4. Other methods.

These were the basic categories, but our analysis, which was driven by the keywords in the title, abstract and author’s keywords that provide information about the nature of the work, was open to the addition of new categories if the keyword clustering suggested that this was pertinent. As will be seen later, this was indeed the case. Specifically, we realized that a large percentage of works were classified into the category “others”. Upon
analysing the clustering of meaningful terms, two groups containing a considerable amount of works were identified. It was therefore considered appropriate to add two new classes to the classification presented above. The taxonomy resulting from this modification will be presented in the following section together with the rest of the mapping outcomes. It should be noted that it is not exceptional to encounter publications that present two or more issues, such as a new mathematical technique and its application to one or several well-known models, or a new or enhanced model and its experimental application. In these cases, our classification chose what we considered to be the most relevant contribution of the various issues presented as the key for the categorization. This was done because we considered that it would be more difficult to sort these works into more than one category. Once the classification task had been accomplished, the results obtained from the systematic mapping were analysed in order to answer the research questions.

Among the outcomes of this SMS a research gap has been identified regarding SB-SRM for software product quality management in industrial environments. This need to be deeply investigated in a SLR developed with such purpose.

### 3.2.2. Results of the SMS

**Step 5: Map building** The most obvious fact to emerge when looking at the final set of primary studies is that the majority of them (77%) are from conference proceedings and only 23% are from journal articles. Journal papers had principally been published in the Journal of Systems and Software (23 papers), and IEEE Transactions on Reliability (19 papers). Next are the IEEE Transactions on Software Engineering which contributes with 9 studies, and the Reliability Engineering & Systems Safety journal with 8 primary studies. The remaining papers appear in various journals but with no significant amount of instances. With regard to the main conferences, as shown in Table 8 the International Symposium on Software Reliability Engineering is the major source of the primary studies included. With regard to the main research groups in this topic, and considering the main author as an index, we found that there were 256 different main authors in the 503 studies selected. Of these, only 18 had published more than three works during the time span analysed.

Upon analysing the geographical distribution it can be noted that China, followed by the USA, are the countries in which most scientific production has taken place as regards SR (see
Moreover, if we consider the contribution of the Pacific Rim area, the percentage of the total amount of works included rises of up to 76%. With regard to the evolution in the number of papers published, we found that the amount of activity in this topic area has increased in recent years, particularly from 2005 on. However, since 2012 the number of papers published on this topic appears to have drastically decreased. Figure 3.2 shows the publication frequency from 2003 until 2014.

If attention is now paid to the nature of the work, it will be noted that, as is shown in Figure 3.3 and Table 3.7 the majority of the primary studies selected claim to present new or enhanced models. With regard to the origin of the papers, that is, whether they have been produced in an academic or industrial environment, Fig. 1 shows that most of them appertain to an academic environment. Indeed, only 40 papers of the total of 503 papers included originated from an industrial environment, i.e., only 8% of the total. If we focus on those works that are of our most interest to us, i.e., those 318 regarding Reliability Models, it can be highlighted that most modelling efforts are in a predictive area: 216 of the selected...
primary studies are focused on predicting reliability as opposed to just 102 that deal with assessment methods and models.

![Figure 3.3: Nature of the Work](image)

<table>
<thead>
<tr>
<th>Nature</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>298</td>
</tr>
<tr>
<td>Technique</td>
<td>95</td>
</tr>
<tr>
<td>Case Study</td>
<td>20</td>
</tr>
<tr>
<td>Discussion</td>
<td>51</td>
</tr>
</tbody>
</table>

Table 3.7: Nature of the Work

The various works that propose a reliability model have been initially classified according to the SRM taxonomy of [33] presented in step 4 in this section. However, during the study we found up to 141 works that did not fit into any of the three specific classes, signifying that approximately 44% of the selected papers had to be classified as "others". This option was discarded for two reasons:

- The percentage of papers is sufficiently representative of the total to be considered as more than simply "others".

- We have used keyword clustering to identify that a significant number of them, 89, could fit into new classes.

The way in which the SMS methodology runs the keyword clustering, or grouping of meaningful terms, is fairly simple. First we extract a list of the most meaningful terms or "keywords" and then we identify the groups or "clusters" of these keywords. For example, in the IEEE Taxonomy terms or keywords such as back-propagation, machine learning, statistical learning, greedy algorithms, support vector machines, particle swarm optimization, genetic algorithms or fuzzy logic (among others) are classified in the Computational and Artificial Intelligence category. In a similar way, as we noted that a significant number
of works that had initially been classified as "other" contained these kinds of terms, we concluded that it would be appropriate to define a separate category for them. As a consequence of this process we decided to add two more classes to the classification provided by [33] because, from our point of view, the new taxonomy would better reflect the reality of the papers under study. These classes are those which correspond to the models inspired by AI based techniques and models built on the consideration of the Static structure and Architecture of the product, namely Static and Architectural Reliability models. We have also slightly modified the scope of the BBN class to include non-SRG models and those that are in some way related to the Bayes Theorem and bayesian inference other than the solely Bayesian Belief networks, and we consequently renamed this class as Bayesian Methods.

The following new taxonomy was therefore obtained:

1. Software Reliability Growth Models (SRGM).

2. Bayesian Methods (BYM).

3. Test-Based Methods (TBM).

4. Artificial Intelligence based techniques (AI based).

5. Static and Architectural Reliability models (SARE).

6. Other methods.

The well-known SRGMs are those time based models that, by means of an empirical approach first used in hardware reliability, describe how the underlying defects affect the observation of failures. The BYM are methods that, by means of a Bayesian Inference, account for the influence or effects of one event on another. These are often implemented as Bayesian Belief Networks and are used for modelling by combining disparate information. The Test Based Methods are specifically interested in considering the way in which the software is used. The selection of test cases should be based on the software’s Operational Profile which reflects the software’s input space. AI based methods are those inspired by Artificial Intelligence techniques, typically Neural Networks, but also other automatic learning or reasoning concepts. Finally, SARE models are built on both the consideration of the architecture
and the static features of the software, and correlate software engineering measures and Software Reliability. Such models are intended to be used during the early stages of the Life Cycle, before the actual software is available. We also have a class for "other" methods and models that contains ad hoc techniques based on particular theories, such as Reliability Block Diagrams, Risk Analysis, Failure Mode Analysis, the Deterministic Chaos Theory and the Cloud Model Theory, or which were developed for very specific cases, such as an ad hoc model for a particular N-Version programming system or methods based on engineering metrics and benchmarking or early software reliability assessment, based on software behavioural requirements. The classification of the papers based on our taxonomy is presented in Figure 3.4 and Table 3.8.

![Pie chart showing classification of previous works](image)

**Figure 3.4:** Classification of previous Works

<table>
<thead>
<tr>
<th>Class</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRGM</td>
<td>145</td>
</tr>
<tr>
<td>BYM</td>
<td>17</td>
</tr>
<tr>
<td>AI based</td>
<td>54</td>
</tr>
<tr>
<td>SARE</td>
<td>29</td>
</tr>
<tr>
<td>TBM</td>
<td>8</td>
</tr>
<tr>
<td>Others</td>
<td>45</td>
</tr>
</tbody>
</table>

**Table 3.8:** Classification of previous Works

Table 13 provides details on the percentage of proposed models by year and class. This has allowed us to obtain that there is a similar tendency in all the years studied. In general, the type of model most frequently used is SRGM, whereas those least used are TBM and BYM.

Figure 3.5 provides a graphic representation of the evolution. As can be seen, although the classical SRGM approach is still the most representative, it would appear that the other techniques are gradually gaining on the SRGM approach, although they are less
representative. It is also important to note the significant appearance of other approaches such as the Artificial Intelligence and the Static and Architecture based approaches, which have grown rapidly since they first appeared.

We have also created a Keyword cloud using the various terms and keywords that recurrently appear in association with the SRM topic, in which the most frequently used terms are written in larger letters, thus showing on which areas the works are focused, see figure 3.6. Having obtained all the results, in the following section we shall answer the questions posed in step 1, table 3.2

3.2.3. Answering the research questions.

This mapping study reports on the high level aspects being researched as regards the modelling of reliability in software systems. This research has been conducted in a systematic manner and with minimal bias through the formulation of a set of research questions which, having achieved the mapping, can now be answered.

RQ1: How many, who carried them out and what kinds of SRM activities have there been in the past 11 years?

Our results as per figure 3.2 suggest that activity in the topic is steadily increasing, although
the anomalous data for 2012 and 2013 need further analysis to assess whether a real change in trends has occurred. We do not believe that this is the case, but rather that other reasons such the replacement of key terms with other emerging terms might better explain this anomaly, i.e., the term "dependability" is now sometimes used with a similar sense to "reliability". As indicated previously in figure 3.7, the initiatives on SR modelling are largely led by academic teams principally located in the Pacific Rim area, with an important prevalence of China, Japan and Taiwan in addition to the notable contribution of the USA. It is also interesting to point out that the majority of industrial proposals, figure 3.8, have been proposed in the USA and Europe. It can therefore be highlighted that there are differences between the countries that lead research depending on the environment.

Figure 3.6: Software Reliability frequently associated Concepts

Figure 3.7: Academic Works Distribution

We should also note the high dispersion of working groups, 256 for 503 papers, in addition
to the fact that only 18 of the 256 main authors had signed more than three works, which has led us to consider that many of the works published have no real continuity and that their development ends just after being published. The low concentration of publishing forums (note that the main contributions in either journals or conferences consist of barely twenty papers), together with their heterogeneity is also noteworthy. All of the above can be interpreted as signs of the discipline’s immaturity.

**RQ2: What SRM research topics have been addressed over the past 11 years?**

The first conclusion is the still important prevalence of SRGM, table 3.8. It is also important to note that 65% of the works studied (Figure 3.3) deal with either (1) a proposal for a new model or an enhancement of previous ones or (2) the application of one or several of them. What is more, most of the works still focus on the classical black-box statistical approach. It is both important and interesting to point out that this study shows that Artificial Intelligence techniques and static architecture based models are perhaps emerging as the next step in this topic’s evolution. Another relevant piece of data is that the third class related to number of studies is ‘others’, which reflects enormous heterogeneity, and is perhaps a sign of a lack of maturity, not as regards the topic itself but as regards the definitions of its generalized and accepted foundations. Please recall that this class typically accommodates adhoc models that are only applicable in particular circumstances, or models based on very particular techniques. It is also worth highlighting that the authors were surprised to find so few modelling activities based on normative proposals such as international standards like the [60] ISO/IEEC 25010. It is of great importance to carry out further research on this in order to avoid the problem of a lack of consensus on this topic’s foundations.
RQ3: Which models are in use?

Our study revealed (Figure 3.3) that very few real-world Case Studies (CS) have been published. The number of CS is marginal when compared to the total number of papers presented. Although it is increasingly common to find that theoretical proposals are completed with application examples, it is often the case that they cannot be considered as real CS. We have also observed that only the 8% of the total number of papers included originated in industrial initiatives. Upon analysing these works we noted that many of them are empirical ad hoc modelling or lesson learning reports on very specific reliability analyses. We further noted that SRGM are the most frequently applied or analysed in industrial environments (roughly a third). There are three examples of industrial initiatives that apply BBN, and two ANN applications. The remaining works (the last third) are either discussions on a variety of reliability related subjects, e.g. data quality and the customers’ viewpoint, or are the automated generation of test cases which are proposals that are based on, for example, exotic theories such as the Chaos Theory, Risk analysis or SFTA, in addition to several comparative studies. From all of the above it would appear clear that a barrier preventing the generalized use of the SRM proposal in industry still exists.

RQ4: Which terms are most often associated with Software Reliability Modelling?

As mentioned previously, the diversity of terms is important and represents an obstacle to the achievement of the maturity of this topic. This phenomenon is illustrated in the weighted cloud presented in figure 3.6, in which it is possible to observe the density of terms, although those most frequently used are: program testing, software fault tolerance, software metrics and stochastic processes. It is also worth to mention how tiny "software reliability model" and "software reliability modelling" appears on the above graphical representation.

An Additional outcome, partially linked to our third research question, is that our mapping study has unveiled a possible gap in research since no significant work appears to have been carried out as regards modelling reliability by following the proposals in international standards such as ISO/IEC 9126 [62] or the ISO/IEC 25K family [59, 60]. The next step will be, then, to extend the study by using a Systematic Literature Review involving more specific topics, as our position is that the issue of the applicability of the models to real-life environments needs to be addressed. As we also believe that standard inspired reliability
modelling will play an important role in achieving the "real life" goal, this SLR will seek software reliability models based on standards. The final goal will be to propose a Reliability model that is, as far as possible, based on standards and which will then be applied to real systems.

3.3. THE SYSTEMATIC LITERATURE REVIEW ON SB-SRM

This section reports on the Systematic Literature Review on Standard Based Software Reliability Modelling research conducted with a twofold objective. First of all it is aimed to complete the necessary State-of-the-Art on such topic. The second objective will be to draw some conclusions on how to tackle our main objective, which is applying Standards in a industrial environment. In this effort we will take into consideration for this research on Software Product Quality the International Standard proposals such as those from ISO/IEC or IEEE and other like MIL-STD or ECSS for specific industrial environments. We have chosen the ISO/IEC 25000 "Software Product Quality Requirements and Evaluation" (SQuaRE) series of standards as a reference framework for this work. The rationale behind this selection is that the International Standards, and SQuaRE in particular, tackle the well-known lack of consensus and the variety of views on what Software Quality is bringing together the efforts of hundreds of volunteers representing varied viewpoints and interests but also SQuaRE is the most recent release on this field and then, arguably, offers the more mature proposal in the framework of Standard Based Software Product Quality.

The Quality Models provide a framework with which to collect stakeholder needs. In the SQuaRE proposal [59, 60] the Quality of a system is understood as the degree to which the system satisfies the stated and implied needs of its various stakeholders. It is, thus, necessary to consider Quality from different stakeholder perspectives. Among the documents in the ISO/IEC standard the 25010 "Quality Model" [60] defines a product quality model composed of eight characteristics which are further subdivided into sub-characteristics (Figure 3.9). This model is understood as a structural model that SQuaRE defines as; "Quality model: defined set of characteristics, and of relationships between them." Thus in this research modelling means analyzing the components and their relationships instead of reducing the
observable phenomena into a set of mathematical formulae. The standard also provides definitions of both each quality characteristic and the sub-characteristics.

This Hierarchical Decomposition strategy is intended to provide a means to deal with the conceptual complexity in order to describe Quality as a multidimensional property, in addition to providing the means to obtain simple measurable attributes that are suitable for further combination into a quality index.

The research object: This research focus on Software Product Reliability. The Software Product is understood as the whole set of source code, design and development documents and software operation manuals. Reliability is, on the other hand, a broad concept that we apply whenever we expect something to behave in a certain way. Very different definitions for this concept have been proposed, from the classic hardware legacy definition in terms of probability of failure-free operation or the vision of Reliability as continuity of correct service to the more recent proposals that extend the vision of reliability in order to integrate the user’s perception of the system. This lack of consensus represents a major handicap and is a symptom of a topic still on development and somehow immature. The International Standards like those from IEEE and ISO/IEC organization represent a very valuable effort.
on the way to gain such a common agreement.

In the ISO/IEC SQuaRE context Software Product Reliability is defined [60] as the degree to which a system, product or component performs specified functions under specified conditions for a specified period of time. It is to be noted that this vision is broader than the classic one as probability of failure-free operation. Reliability is considered, then, as a combination of Availability, Maturity, Fault Tolerance and Recoverability concepts for which a formal definition, detailed in table 3.9, is also provided although the nature of such a combination is not defined in the Standard.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time.</td>
</tr>
<tr>
<td>Availability</td>
<td>Degree to which a system, product or component is operational and accessible when required for use.</td>
</tr>
<tr>
<td>Maturity</td>
<td>Degree to which a system meets needs for reliability under normal operation.</td>
</tr>
<tr>
<td>Fault tolerance</td>
<td>Degree to which a system, product or component operates as intended despite the presence of hardware or software faults</td>
</tr>
<tr>
<td>Recoverability</td>
<td>Degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system</td>
</tr>
</tbody>
</table>

Table 3.9: SQuaRE Reliability Characteristics definitions

The fact that the Standard does not discuss how these concepts relates one each to the others is a relevant hindrance to the applicability of the proposal as well as could be masking whether some similar concept is missing in the proposed description. It is our understanding, thus, that the relationship between those proposed components need to be analyzed and so we will do on section 5. It is also necessary to note that, reliability can and should be considered from different points of view in order to meet the different stakeholders’ needs. The SQuaRE proposal recognizes different stakeholders: Primary user, Secondary users and Indirect user as well as their different needs, e.g. in the 25010 document [60] we can read, table 3.10, the following differences:

Despite the fact that these are two extremely relevant points to consider the standard’s documents provide little information on how to deal with them. Therefore, this research is very interested on investigate how both, academia and industry, have faced these issues; how the proposed sub-characteristics relates one each other and how the user’s needs are
3.3. THE SYSTEMATIC LITERATURE REVIEW ON SB-SRM

### Table 3.10: SQuaRE Reliability Views

<table>
<thead>
<tr>
<th>USERS</th>
<th>Primary user</th>
<th>Secondary user</th>
<th>Indirect user</th>
</tr>
</thead>
<tbody>
<tr>
<td>User needs</td>
<td>Interacting</td>
<td>Interacting</td>
<td>Using output</td>
</tr>
<tr>
<td>Reliability</td>
<td>How reliable does the system need to be when the user uses it to perform their task?</td>
<td>How reliable does updating the system with new content need to be?</td>
<td>How reliable does the output from the system need to be?</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Captured, since they will greatly determine whether or not the proposal will result in an applicable schema.

#### 3.3.1. Data gathering for the SLR

The present Systematic Literature Review involved three software engineering researchers, two of whom acted as primary screeners and the third of whom acted as an auditor. The Review was conducted by applying the replicable process presented in Chapter 2. The search for primary studies was conducted in February 2014 to cover papers published between 1991 and 2014. This time span was chosen as a trade-off between feasibility and relevance, taking into account the release date of the ISO/IEC 9126, which is the antecessor of our chosen framework. In terms of the scope of the research, our interest lies in the software product, leaving aside the analysis of software production processes, and the variety of techniques used to build reliable software.

Our systematic review aims to, as far as possible, identify and analyze those primary studies that propose Software Product Reliability Models, theories or industrial experiences based on or closely related to International Standards. We have approached this aim by answering the following research questions (Table 3.11):  

<table>
<thead>
<tr>
<th>WHICH</th>
<th>Q1. Which software reliability models have been developed by following the recommendations in International Standards?</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOW</td>
<td>Q2. What are the experiences as regards determining reliability on the basis of International Standards?</td>
</tr>
</tbody>
</table>

**Table 3.11: SLR Research Questions**

The first question Q1 aims to identify which relevant related work is available on the topic
in which we are interested, i.e. quality or reliability models developed as a refinement of an International Standard proposal or adhering to its methodological recommendations or strategies. Q2 investigates how the proposed models have been or can be applied to either academic research or real industrial projects. This question also aims to investigate the difficulties involved in such an application. The ultimate goal is to obtain an insight into how to apply Software Product Reliability Modelling to industrial environments based on representative International Standards, paying particular attention to the impact of SQuaRE and ISO/IEC 9126 \[62\] series.

The research questions and scope were used to derive a set of relevant keywords which has been further split into two sets related to domain specific concepts and generic contextual terms (Figure 3.10).

![Figure 3.10: Research Keywords.](image)

These keywords have been used to build the research strings used to retrieve primary studies form the available repositories. See Appendix I for details on the search strings and queries.

Inclusion and exclusion criteria, which are now formalized on table 3.12 and 3.13 for further application.

The search was conducted using automated search engines of three of the largest and most complete scientific databases: IEEE Xplore www.ieeexplore.ieee.org, ACM Digital Library www.portal.acm.org and Science-Direct www.sciencedirect.com that were chosen because are widely recognized \[18, 39\] as being an efficient means to conduct Systematic
3.3. THE SYSTEMATIC LITERATURE REVIEW ON SB-SRM

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC1</td>
</tr>
<tr>
<td>IC2</td>
</tr>
<tr>
<td>IC3</td>
</tr>
<tr>
<td>IC4</td>
</tr>
<tr>
<td>IC5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC1</td>
</tr>
<tr>
<td>EC2</td>
</tr>
<tr>
<td>EC3</td>
</tr>
<tr>
<td>EC4</td>
</tr>
<tr>
<td>EC5</td>
</tr>
<tr>
<td>EC6</td>
</tr>
</tbody>
</table>

Table 3.12: SLR Inclusion Criteria

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC1</td>
</tr>
<tr>
<td>EC2</td>
</tr>
<tr>
<td>EC3</td>
</tr>
<tr>
<td>EC4</td>
</tr>
<tr>
<td>EC5</td>
</tr>
<tr>
<td>EC6</td>
</tr>
</tbody>
</table>

Table 3.13: SLR Exclusion Criteria

Reviews in the context of Software Engineering. We applied search strings based on the logic combination of the keywords identified above. As a result of this step, we obtained a total of 1820 papers: 423 from ACM digital library, 1161 from IEEE Xplore and 236 from Science Direct.

After the initial data preparation, the selection process comprised four iterations. The first was carried out separately by two reviewers while the following one was conducted by one evaluator. The intention of this step was to reduce any bias in the results. The last two iterations were conducted as a joint review by the three researchers. Discrepancies regarding the inclusion and relevance of selected paper were discussed and agreed upon. Table 3.14 summarizes the figures of the selection process.

<table>
<thead>
<tr>
<th>Review step</th>
<th>Rational</th>
<th>Works at output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automated search</td>
<td>Results sought in DBs</td>
<td>1820</td>
</tr>
<tr>
<td>Data preparation</td>
<td>After duplicates rejection (1653) No paper, errors, hardware (1406) Not applicable fora (801)</td>
<td>801</td>
</tr>
<tr>
<td>Independent review</td>
<td>Review Title and Abstract, rejecting those out of scope or not relevant to our aim.</td>
<td>32</td>
</tr>
<tr>
<td>Auditing</td>
<td>Proposed papers review resolving discrepancies and rejecting those not relevant to our aim.</td>
<td>29</td>
</tr>
<tr>
<td>1st joint review</td>
<td>Paper’s contents joint review</td>
<td>26</td>
</tr>
<tr>
<td>2nd joint review</td>
<td>Final decision on input to Thematic Synthesis</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 3.14: The Selection Process
The first noticeable fact is possibly the important reduction in the number of papers from almost two thousand down to first thirty-two candidates which underwent an in-depth analysis. This is actually owing to the search strategy itself, since we performed a broad general search but were interested in a narrow research scope, hence the dramatic decrease in the amount of papers.

3.3.2. Results of the SLR

With regard to the results of the first review, the two independent researchers reached a high level of agreement. Of a total of 32 possible hits, 23 of these were selected by the two reviewers, 5 by only one and 4 by the other. A third researcher then reviewed this result with the purpose of audit selection and resolving discrepancies. A joint review agreed a first group of 26 works to be analyzed in depth. This review was conducted by a researcher and presented in a second joint review from which we obtained the 11 papers, on table 3.15, that form the basis of the thematic synthesis. It is worth noting that nine of these were in the initial set of 23 papers chosen by both screeners. The classification had an agreement of 71.9%. The obtained kappa value was 0.85, which is a fairly good agreement.

In order to help us conduct this analysis in a systematic and structured manner, we chose an existing [120] classification of research approaches in addition to the research questions. First of all we analyzed what kind of research was performed. This data (Table 3.16) helped us to understand what activities are being carried out in addition to providing a first insight into the impact of SB-SRM in academia and industry.

It is interesting to note that almost all the evidence gathered falls into two kinds of research: “Proposal of a solution”, that is, works that propose a solution and argue for its relevance without a full-blown validation; and "Evaluation", which corresponds to papers that investigate a problem in practice. Our own work is research consisting of the proposal of a solution as defined in the aforementioned reference.

Then we focused on what the selected works deal with. At a higher level we concentrated on whether the work is Reliability specific or it dealt with in a more general scope such as that of Dependability [6, 75] or Software Product Quality. The results in 3.11 show that when in the context of Standards, Software Reliability is often considered at a high level
### Table 3.15: The Selected Works

<table>
<thead>
<tr>
<th>Work</th>
<th>Title</th>
<th>I.C.</th>
<th>Year</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Software dependability evaluation based on quality characteristics</td>
<td>1,2</td>
<td>2010</td>
<td>[126]</td>
</tr>
<tr>
<td>2</td>
<td>Measurement of software requirements derived from system reliability requirements</td>
<td>1,4</td>
<td>2010</td>
<td>[110]</td>
</tr>
<tr>
<td>3</td>
<td>A comprehensive code-based quality model for embedded systems</td>
<td>1,2,3</td>
<td>2012</td>
<td>[83]</td>
</tr>
<tr>
<td>4</td>
<td>Using dependability benchmarks to support ISO/IEC SQuaRE</td>
<td>1,2,3</td>
<td>2011</td>
<td>[47]</td>
</tr>
<tr>
<td>5</td>
<td>Predicting Quality of O.O. Systems using a QM based on design metrics &amp; data mining</td>
<td>1,3</td>
<td>2009</td>
<td>[26]</td>
</tr>
<tr>
<td>6</td>
<td>Experience with the use of IEEE 982.1 standard in software programs</td>
<td>3</td>
<td>1997</td>
<td>[117]</td>
</tr>
<tr>
<td>7</td>
<td>Handbook-based high unit-value software reliability prediction method</td>
<td>4</td>
<td>2010</td>
<td>[1]</td>
</tr>
<tr>
<td>8</td>
<td>A process framework for customizing software quality models</td>
<td>1,2</td>
<td>2007</td>
<td>[80]</td>
</tr>
<tr>
<td>9</td>
<td>Analysis of contribution of conceptual model quality to software reliability</td>
<td>4,5</td>
<td>2010</td>
<td>[38]</td>
</tr>
<tr>
<td>10</td>
<td>Software quality and CASE tools</td>
<td>4,5</td>
<td>1999</td>
<td>[49]</td>
</tr>
<tr>
<td>11</td>
<td>Software Reliability Measurement. Use Software reliability Growth Model in testing</td>
<td>4</td>
<td>2005</td>
<td>[53]</td>
</tr>
</tbody>
</table>

together with Quality or Dependability. Regarding which particular Standards are being reported in the selected papers. Table 6 shows that the SQuaRE series still has a very limited impact since most of the available works reference its antecessor, ISO/IEC 9126 [62]. Others proposals like IEEE-std 982 [58] and IEEE-std 1633 [56] also have relevance in this area and are sometimes cited together with the ISO/IEC 9126.

In each paper we have also identified whether SQuaRE’s Reliability characteristics are treated on any way or not. We have also search for the Reliability characteristics considered in other relevant standards, e.g. Robustness as defined in the IEEE-Std 610.12 [57]. We have found that there is little analysis of sub-characteristics, and in most cases the papers do not in fact go beyond the first level of decomposition, associating measurable attributes directly with the high level characteristic. Finally, we identify how the hierarchical decomposition and the measure aggregation are treated. The measurement process is only rarely explicitly detailed and the literature sometimes even refers to the values of attributes not explicitly defined in the document.
Only one of the works analyzed deals with the Operational Profile [86], which should play a central role in reliability assessment. It is also interesting to analyze whether the definition of reliability requirements is taken into consideration within the modelling frameworks, and it is worth noting that this is often not the case.

### 3.4. THEMATIC SYNTHESIS

The objective now is to build the answer to our research questions. In order to answer our "which" question we focused on which models or modifications to models are proposed,
and which aspects related to Reliability factors are dealt with. We answered our "how" question by searching for reports on models applied to real industrial projects or academic experiments and analyzing how the concept of reliability is treated and understood when in the SB-SRM context, in addition to the specific techniques that are applied.

**Answering the which:** The most common approach is that of taking the Standard’s proposal as a starting point for a model proposal. Such as in [1] where the author states that the proposed model is constructed on the basis of both ISO 9126 and other similar models like those of McCall, Boehm or FURPS. However, no explanation is provided as to how it is developed beyond adherence to the strategy of hierarchical decomposition. The Proposed model is further empirically analyzed in the context of the Object Oriented paradigm.

In [126] the authors present a method for evaluating reliability/dependability (the two terms are used interchangeably) with reference to the quality model of ISO/IEC 9126. The use of the standard is limited to guiding the selection of the quality index. They do not propose measures. However, a measure aggregation proposal is presented by combining weights with the use of techniques from Fuzzy Theory and the Analytical Hierarchical Process.

In [47] the authors address the problem of integrating COTS software into the product and the need to ensure global quality, focusing on Recoverability, a sub-feature of Reliability, and how SQuaRE deals with this problem. These authors propose enriching the ISO/IEC 25045, thus by extension SQuaRE, integrating the concept and techniques of dependability benchmarking in order to tackle the limitations in the evaluation of software products in the presence of generic disturbances. In [26], the conventional determination of Reliability is shown in addition to how parameters can be applied on the basis of standard based models to estimate participation in computing the Reliability of missing data such as the initial design fault density.

The complexity of the Reliability concept, which is characterized by its multiples facets, is shown in the majority of the works on this topic but several works [47, 83, 110, 126] specifically tackle the conceptual analysis of software reliability and the variety of related concepts proposed in the different standards. In [110] the authors analyze the variety of reliability views and propose to harmonize the treatment in different standards (ECSS ¹).

¹ECSS stands for European Cooperation for Space Standardization
ISO9126 and IEEE-stds) into a generic standard-based reference model in which reliability is expressed as a set of functional requirements and is therefore implementable and measurable. This is intended to ease the application of reliability engineering and also contributes to filling the gap between the upper abstraction levels in the conceptual decomposition and the technical attributes at lower levels. There is some available evidence regarding the impact that a variety of factors have on the Reliability delivered. The influence of the production process has been analyzed in several of the selected works, usually by means of survey studies. In [49] the authors examine the impact of both back-end and integrated CASE tools on the quality, and hence the reliability, of the software product using the ISO/IEC 9126 quality definition to drive the survey research. In [38] the guidance of ISO/IEC 25010 is used to analyze how the quality of conceptual models influences software reliability and which factors of conceptual model quality have an effect on software reliability.

**Answering the how:** A first finding when considering how to apply SB-SRM is the need to tailor the model to the particular needs in a case by case manner despite what the standards do not provide details on how to proceed and most of the analyzed works merely show the final tailored model. However, survey research is reported in [80] that seek to develop and validate a new process framework with which to customize software quality models in order to meet organizational goals which is a requirement for the application of generic quality models. The author proposes to convert the software Quality Model into a survey questionnaire as it is also proposed in [49]. The authors of [53] propose using a survey based on ISO/IEC 9126 reliability metrics, later processed following the Analytical Hierarchical Process, to request information regarding customer satisfaction in order to guarantee that stakeholders’ expectations are met.

Another basic task to be dealt with is the selection of measures. SQuaRE does not provide a specific method to guide this activity. However, in [117] Farr reports the application of the IEEE 982.1 standard to two industrial projects. The role of the standard was to provide guidance for the measure selection process.

In [26] the authors report the development of a quality model oriented to the internal quality of embedded source code which covers requirements for the source code. The model is hierarchical in nature and integrates the vision proposed in the 25010 standard. The rational
to dismiss the classical strategy of hierarchical decomposition and instead of refining the
goal characteristics selecting first the low level source code quality features to then
integrate the quality framework are the difficulties to make the link between the very high
level nature of the proposed model on the standards and the source code properties and
so with the implementation. It is worth to note that such is the problem that [110] intend
to address from a different perspective based on the operationalization of the high level
quality requisites. As mentioned in most cases the papers do not go beyond the first level of
decomposition, associating measurable attributes directly with the high level characteristic.
The aggregation strategy for the low level attributes is a central issue when using hierarchical
decomposition as a basis for modelling, despite which it is only occasionally dealt with
in literature. In many cases the proposed aggregation of measures consists of a more or
less sophisticated weighted sum. In [126] the researchers present a measure aggregation
strategy based on the combination of subjective and objective weights to determine the
final weight. The need to define and use an operational profile for the determination of
reliability is well known, but only one of the selected papers [47] mentions this. This is
almost certainly because the operational profile is more closely related to both the fault
process and the testing activities. A summary of the main subject of each of the analyzed
works, in terms of our "which" and "how" research questions, is presented in Table 3.17. It
is to note that some of the studies deal, also, with issues out of the scope of our research.
Those issues are not detailed on our synthesis.

3.5. STATE-OF-THE-ART WRAP-UP

In the following we summarize the main findings from the previous comprehensive literature
review as well as and discuss possible limitations to our stated conclusions. The mitigation
actions to each of the possible threats to the validity of our analysis are also detailed.

3.5.1. Threats to validity

Despite the fact that both studies, SMS and SLR, have been carried out by following a formal
methodology, there may be some threats to its validity. The principal limitations concern
the limited access to sources, a circumstance that may have led to a bias in the selection
of publications owing to the possible existence of studies of interest in other databases.
### Table 3.17: Answer to the Research Questions

<table>
<thead>
<tr>
<th>WHICH</th>
<th>HOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 A model proposal including measure aggregation method</td>
<td>Taking the standard as starting point and applying Fuzzy Theory and AHP</td>
</tr>
<tr>
<td>2 Conceptual analysis of Reliability</td>
<td>By means of an operationalization strategy based on treating Reliability as a functional requirement.</td>
</tr>
<tr>
<td>3 A model proposal.</td>
<td>Hierarchical structure integrating ISO/IEC 25010</td>
</tr>
<tr>
<td>4 SQuaRE model completeness analysis</td>
<td>Enriching/Extending the proposal</td>
</tr>
<tr>
<td>5 Reliability assessment</td>
<td>Estimating missing data from Intl. Std.</td>
</tr>
<tr>
<td>6 Report of application</td>
<td>Measure selection guided by Intl. Standard</td>
</tr>
<tr>
<td>7 A Model proposal</td>
<td>Taking the standard as starting point</td>
</tr>
<tr>
<td>8 Capturing stakeholder’s expectations</td>
<td>Converting the Quality Model into a survey questionnaire</td>
</tr>
<tr>
<td>9 Impact of factors on delivered Reliability</td>
<td>By means of a survey study</td>
</tr>
<tr>
<td>10 Impact of factors on delivered Reliability</td>
<td>By means of a survey study</td>
</tr>
<tr>
<td>11 Capturing stakeholder’s expectations</td>
<td>By means of a survey questionnaire based on ISO/IEC 9126</td>
</tr>
</tbody>
</table>

However, the databases used cover the area of software engineering well [18, 39] and we have no reason to believe that this does not apply to software reliability. We are, therefore, reasonably confident that we are unlikely to have missed many significant published studies.

Some relevant papers might not have been found in the digital databases when using our search and selection protocol. Automated searches rely on both search engine quality and how researchers write their abstracts. Although we are reasonably confident as regards how well digital databases classify and search indexed work, if abstracts and keywords are of poor quality it is clear that the search will be greatly flawed. However, since our selected keywords are commonly used terms the possibility of any significant contribution not mentioning these key words in the title or abstract is minimal.

Another threat lies in the difficulties involved in selecting and classifying the primary studies selected owing to the lack of an appropriate taxonomy and the wide diversity of terms used in the topic of interest. This may also have led us to miss some relevant studies since the search string used in the SMS consisted of the terms "Software Reliability". However, since this is, on the whole, the most commonly used term, the possibility of any significant contribution not mentioning the words "Software Reliability" in the title, abstract or keywords is minimal.
With regard to the classification, we have added new categories to the existing taxonomy, and we therefore believe that the classification of the papers is mostly accurate.

Another threat that needs to be considered is the possibility of bias in selection. This concern was addressed as described in section 3.2. During the SMS, as mitigation action the selection phase was conducted independently by two authors then audited by the third one and finally reviewed in a joint revision. In the case of the SLR only this Thesis author coded all the papers, and some bias could therefore have been introduced in the results for a variety of reasons such as the author’s subjectivity. However, the Thesis co-directors’ supervision as regards reviewing both the included and the excluded papers leads us to believe that this error, if it exists, is minimal.

Finally, only studies published in the English language were selected in the search, signifying that potentially important studies published in other languages have not been considered. However, since English is the most widely adopted language as regards writing scientific papers, the eventual bias owing to this issue is minimal.

It is also possible that new relevant works have been published since we closed the collection of the primaries studies included in our present analysis however we consider our proposal yet valid because uses the last standards proposals and has been developed working in a methodical way to propose results that, as we will see further, can be validated in real-world set-up.

3.5.2. Conclusion

The principal results attained have allowed us to observe that there are many research groups publishing very different initiatives for Software Reliability modelling using a variety of approaches and points of views. The lack of consensus on what Software Reliability is a well-known fact, and it does not appear to have evolved towards an increased agreement over the last decades, on the contrary the variety of definitions has increased.

We have attempted to use a classification proposed in literature but have found that it was not sufficient to classify the papers selected, and we have therefore proposed a new taxonomy where, surprisingly, the two new classes were the second and the third as regards
those in which most proposals were found (after the classic SGRM approach).

The following new taxonomy was therefore obtained:

1. Software Reliability Growth Models (SRGM).

2. Bayesian Methods (BYM).

3. Test-Based Methods (TBM).

4. Artificial Intelligence based techniques (AI based).

5. Static and Architectural Reliability models (SARE).

6. Other methods.

The high dispersion of working groups, 256 for 503 papers, in addition to the fact that only 18 of the 256 main authors had signed more than three works has led us to consider that many of the works published have no real continuity and that their development ends just after being published. We interpret these facts as signs of the immaturity of the Software Reliability Modelling topic in which the first controversial issue is the variety of definitions the available literature offers for the study object. This lack of consensus represents a major handicap to the development of effective models.

Another important point that should be noted is that few real-world case studies have been published, and most model application works have, to date, focused on academic experiments. More research is therefore required in order to understand the problems of applying SRM techniques in real-world context. In the same vein we were surprised to find so few modelling activities based on normative proposals such as international standards like the ISO/IEC 25010 [60]. Our study revealed that the number of published Case Studies is marginal when compared to the total number of papers presented.

Deepening the analysis of this point the main outcome from the SLR on software reliability assessment based on representative international standards is the confirmation that Standard Based Software Reliability Modeling (SB-SRM) is receiving limited attention from the academic community in addition to having little impact on industry, or at least industry is
not reporting on its application.

From all of the above it would appear clear that a barrier preventing the generalized use of the SRM proposal in industry still exists. The international standards can play a first order role in order to gain such a consensus and thus ease the industrial application of Software Reliability Modelling, not only by offering guidance for the modelling task itself, but also because they are recognized by industry as a trustable source of recommendation and guidelines.

As introduced in Chapter 1 our main research hypothesis is that SB-SRM can be effectively applied to Software Reliability Analysis and this investigation is about why this is not happening? Some clues can be obtained from the literature reviewed. The complexity of the concept itself is possibly the main impediment to the broad application of Software Reliability Models in an industrial environment, but also that it is necessary to consider reliability from different perspectives in order to meet the different stakeholders’ needs. This point seems to be receiving very little attention although it is to note that it is already recognized in the standards notable in ISO/IEC SQuaRe.

In addition and related to the fact that it is necessary to consider Reliability from different perspectives in order to meet the variety of stakeholders’ needs an important issue that is not always taken into account is that Reliability is also a perception (maybe mainly a perception) on how the system behaves. The different stakeholder needs is thus a point of paramount importance that should be better covered in order to increase the industrial applicability of the proposed models

Another recurrent problem concerns the difficulties involved in relating the low level attributes of the source code or the design documents with the high level characteristics of the system behavior once in exploitation. Related to this is the apparent lack of aggregation strategies for the proposed models. Failing to make this link in a sound, understandable and simple manner implies that the models are not greatly effective in supporting the decision making process that the industry’s stakeholder needs, and are not therefore used.

On the following we will formally characterize those main identified obstacles for a broad application of SB-SRM together with a selected approach to overcome them which will
lead us to propose an innovative layout and framework for Software Reliability Analysis in Industrial environment.
On this chapter a solution to the identified issues is proposed. The solution proposal is made up of a new innovative Software Reliability Model and an application framework. The Model proposal is based on developing the latest normative advances, in particular the ISO/IEC-SQuaRE series of International Standard on Software Quality. The framework is oriented to promote its industrial applicability. It will be developed along the next Chapter 5. The solution proposal presented in this chapter is already published as part of [43] and once again in [42] together with some of the validation techniques applied in this thesis.

*All models are wrong, but some are useful*

George Edward Pelham Box

4.1. STANDARD BASED SOFTWARE RELIABILITY MODELLING

There is a need to assess software reliability, and to be able do so the industry requires models that easily and effectively captures the complexity of this multifaceted concept. Moreover, in the highly competitive industrial context we need means of capturing the user-based dimension of Software Product Reliability, which is important in order to increase end-user satisfaction with the delivered products thus an important competitive advantage. It is also important producing such assessment in a way that can be related to the Product low level attributes and features, the product-based view. The knowledge on such relationship will permit us an informed decision making process close to the user needs and expectations whilst in the production and maintainance phases of the product life-cycle.
As it have been presented in Chapter 3, among the documents in the ISO/IEC SQuaRe family of standards the 25010 about the "Quality Model" defines a Product Quality Model composed of eight characteristics which are further subdivided into sub-characteristics. This model is understood as a structural model that SQuaRE defines [60] as; "defined set of characteristics, and of relationships between them." Thus in this research modeling means analyzing the components and their relationships instead of reducing the observable phenomena into a set of mathematical formulae.

Moreover, industry needs simple models and clear methods that are straightforward to apply in order to be useful to inform the decision making process in a real world context. It is to take to the foreground now that our purpose, as introduced in Chapter 1 is: "To develop a Standard-Based Model for Software Product Reliability description and analysis as well as defining an application framework focused on its Industrial Applicability". Such simple model is to be developed as a consequence of the Literature Review to overcome the unveiled hindrances as described along this Chapter 4. On next Chapter 5 we will develop its application method as a consequence of the Industrial Environment context and constrains.

### 4.1.1. The need for Software Quality Models

A Quality Model is required to knowing, among many others, how to assess the actual reliability level of any software product which is necessary to, e.g. determining the optimal time at which to stop testing and release software as well as to providing data with which to make trade-offs between test time, reliability, cost and performance. Quality Models also [59] provide a framework with which to collect stakeholder needs. Bearing in mind that the fundamental reason for this modeling in Engineering is organizing knowledge to support business decisions. Software Reliability models are then intended to capture the software properties and characteristics in a useful manner in order to support the aforementioned objectives.

Any valuable model have to being founded on sound assumptions, capture the phenomena complexity and be demonstrably accurate, which is usually challenging enough. In industry a valuable model, in addition, have to being practical and cost effective since it has to perform in a real-world context. That means, with a minimum need on resources and a maximum of results. The key to getting Quality Models applied in daily practice is that they must not
only be sufficiently descriptive and simple to apply but must also be able to show that they are profitable to the organization.

The work of Standardization Organizations are good examples of the necessary efforts to achieve such models, although they do not appear to have had a great impact in industry with just a few works on Standard Based Software Reliability Modelling (SB-SRM) and less real world application studies published or at least industry is not reporting on its application, and most model application works having, to date, focused on academic experiments [41] rather than industrial experiences. This is surprising given the relevance of such International Standards in Industry.

4.1.2. Industrial Relevance of International Standards

As it has been discussed along this dissertation, Quality can be an obscure concept at first because what one might see as Quality someone else may not. Hence, the need and purpose of Quality Standards is crucial. Clearly defined standards serve as a framework for businesses making it easier for companies to meet what their consumers consider quality. International Standards [116] are the result of a consensus and are approved by a recognized body and thus Standards provide people and organizations with a basis for mutual understanding, and are used as tools to facilitate communication.

Among the more commonly cited benefits of adopting International Standards are the advantage for consumers since conformity of products and services with International Standards [28, 93] provides assurance to consumers on the quality, safety and reliability of these products and services. For manufacturers, standards are deemed to contribute to rationalize the manufacturing process, reducing the costs and increasing delivered quality. In the same sense it is also cited that [28] standards contribute to speeding up the introduction of innovative products to market and recognized that Standards are an important means of communicating to partners that our products and processes follow recognised good practices thus that they can trust on the Quality of our products.

International Standards play a central role in industry. In particular, the demonstration of compliance with quality standards which is commonly recognized as a gauge of higher performance levels as well generates better positioning in the market by means of increased
customer confidence as is remarked among the key conclusions on [28] when said that International Standards are "important for market acceptance of technology-based innovation".

As a consequence the International Standards [115] have become very popular in industrial context and have stimulated the interest of many companies in their effort to enhance their competitive advantage. ISO 9001 (quality), ISO 14001 (environmental) and ISO 27001 (information security) are examples of standards against which certifiable schemes have been developed. In particular ISO 27001 has become an increasingly adopted international standard due to the growing number of users online and corresponding growth in digital crime. ISO 27001 ensures an organisation’s information is safeguarded from unauthorised access, use, destruction, modification or disclosure. Also, the ISO 9000 series of standards has been adopted [116] around the world for defining and documenting quality management systems for organizations of any size and type.

At one time [93] the International Standards were thought of as being the lowest common denominator, restrictive, and of little importance. That has changed. Today, Standards are recognized as being essential to helping companies be innovative, reduce costs, improve quality and maintain competitiveness in an international marketplace. Companies and organizations increasingly require their providers to comply with International Quality Standards

4.1.3. Identified Hindrances

It is from this perspective that we approach the topic of Software Product Reliability Modelling and have thoroughly investigated the available literature to find out [43] that the main obstacles for a broad industrial application of SB-SRM are related to the inherent conceptual complexity and its associated variety of definitions as well as the lack of methodologies straightforward to apply in a real-world context without failing to capture stakeholder’s expectations and able to relate the software product attributes to the system behavior. In more detail such impairments and our proposal to overcome them, in which the International Standard will play a central role, is presented hereafter.

**Conceptual complexity.** From the evidence collected to produce the State-of-the-Art we have been able to state that the main disadvantage of software reliability models is that
they are too complex to be effectively applied in a daily practice. In order to tackle the issues related to the conceptual complexity and variety of definitions as well as the possible lack of completeness of the proposed models \(^1\) our proposal to mitigate both hindrances is the adherence to mature International Standards guidelines, taking the aforementioned ISO/IEC-SQuaRE International Standard as main source but considering also other Standards to complete SQuaRE where required.

**The stakeholders expectations.** The second major hindrance that we intend to afford is that the existing models are often failing to capture stakeholders’ expectations. To tackle this issue we preconize the view of Reliability as an user oriented concept where perception has a major role. That will suggest us the application of analysis techniques more related with social or economic sciences than the traditional statistic and probabilistic methods which have been severely criticized [44, 73, 74, 76] in the past. In addition current view on reliability has moved from a definition in terms of "probability of failure" to the "degree" or "extent" which also suggest the unsuitability of purely statistic-based analysis.

**Connecting product attributes with system behavior.** Finally, it is also an outcome of our investigation that the existing models are failing to make the link between the very high level of description in the proposed models and the source code properties in a sound understandable and simple manner. In general the Standards do not provide details on how to apply them in a practical realization. This is closely related to the lack of sound and founded aggregation strategies in the standard proposals despite such aggregation procedures are required to translate the product attributes into the higher level Quality characteristics and sub-characteristics. To afford this issue we will make use of well-founded and recognized techniques and simple methods promoting on that way the industrial applicability and the conceptual understanding by the broader group of stakeholders.

*In summary,* our proposed strategy to overcome these aforementioned hindrances will be based on SQuaRe [59, 60] hierarchy layout refurbishment in the context of adherence to International Standard proposal to tackle the general issue of complexity and the explicit treatment of Reliability as user oriented concept in order to capture the variety of stakeholders’ viewpoints. Over that we will afford the analys of the relationships between the

\(^1\)Which is directly related to their complexity
4.2. THE PROPOSAL

As said, our aim is to derive a Software Reliability Model on the basis of existing International Standards, mainly the SQuaRe proposal but if required integrating elements of other similar proposals. We adhere to the hierarchical decomposition strategy to deal with the conceptual complexity although revising the proposed layout in terms of conceptual content of the previously proposed reliability characteristics as well as analyzing its potential lack of completeness and the appropriateness of integrating reliability characteristics from other models.

The conceptual hierarchy We deem that Reliability is a user-oriented concept. In the simplest sense, reliability is an assessment of how well their users think the system provides the service they require. It is also important to consider that different users will have different viewpoints. As mentioned previously professor Musa claims [90] that Software Reliability concerns itself with how well the software meets the requirements of the customer, also in the seminal work on Dependability [6] we read that Reliability is the continuity of a correct service. In the ISO/IEC SQuaRE [60] context Software Product Reliability is defined as the degree to which a system, product or component performs specified functions under specified conditions for a specified period of time. That is, a system will be reliable to the extent that it is available whenever it is required for use and behaves as expected by its users. Wording that in terms of ISO/IEC vocabulary Reliability is, therefore, a certain function of Availability and Maturity in which maturity summarizes the correctness of behavior that is consistent with stakeholders’ expectations.

Maturity means then that user expectations are met and thus no changes or corrections are consequently required, which is Stability as understood on IEEE 982.1 [58] when defining the Software Maturity Index. Fulfilling user expectations also imply a system that permits an easy and flexible use of the facilities, which can be mapped to Robustness in the terms of
IEEE 610.12. This is directly related to the correctness of implementation since Robustness is understood [57] as the degree to which a system or component can function correctly in the presence of invalid inputs or a stressful environment. In summary, Maturity is made up of Stability and Robustness. On the other hand, the Availability of a system depends on how much it fails as regards the effort required to repair it. Availability is thus a certain function of a system’s Fault Tolerance, which determines whether a fault will manifest itself as a failure and the system’s Recoverability, which accounts for the recovery efforts after failure.

The above discussion has been used to derive a new schema, see figure 4.1, for the decomposition of Reliability. This new layout, although being derived from the SQaRe decomposition is innovative by explicitly considering the conceptual hierarchy to establish the relationship between the structural elements. This also permits to make a more complete description considering additional characteristics as well as can also be mapped onto different stakeholders’ needs, thus addressing the issue of capturing different viewpoints and needs. Each level in this proposed hierarchy deals, thus with both a particular description level and a particular user need. We will develop this on the following.

<table>
<thead>
<tr>
<th>RELIABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATURITY</td>
</tr>
<tr>
<td>STABILITY</td>
</tr>
<tr>
<td>ROBUSTNESS</td>
</tr>
<tr>
<td>AVAILABILITY</td>
</tr>
<tr>
<td>FAULT</td>
</tr>
<tr>
<td>TOLERANCE</td>
</tr>
<tr>
<td>RECOVERABILITY</td>
</tr>
</tbody>
</table>

**Figure 4.1: Model Proposal Layout**

**Mapping to the user’s needs** At the top we have Reliability as the user’s perception of the system behavior, this global view being characteristic of end user or higher management levels. Below, the first decomposition levels enable the functional analysis of the global contributors to the user’s perception. The system can be analyzed at this level so as to make decisions regarding the definition of the quality characteristics by considering external constraints and business objectives in a simple manner, i.e. the maturity in terms of updates to the product can be largely impacted by the business model or the recovery delays by extrinsic constrains on operations safety. It is still necessary to operationalize the description to lower levels so as to be able to implement corrective actions or produce an assessment that can be used as input to compute high level indexes.
The first step in that sense is the third decomposition level where still on the basis of Reliability characteristics proposed on International Standards the description is specialized onto secondary characteristics; stability, robustness, fault tolerance and recoverability. This third level enables us to obtain a quantitative estimation of the system behavior, thus external metrics according to ISO/IEC vocabulary. We could, e.g. estimate the Stability by means of the Maturity Index as per IEEE 982.1 proposal or the Robustness using the Breakdown Avoidance while Fault Tolerance coulds be estimated by means of the Mean Time Between Failure and Recoverability by Mean Recovery Time as per ISO/IEC 9126 definitions.

Notwithstanding, the above still do not correspond to those low level software product attributes which are directly observable and measurable, let say lines of code or defects or function points as an example, but to system behavioral assessment. As a consequence, and also on the basis of the discussion presented, our proposal introduces another level below the hierarchy, figure 4.1, defined on the basis of the ISO/IEC characteristics and sub-characteristics.

Making the link  The use of this additional description level is required to maintain the link between the user perception and managerial needs and the engineering levels in charge of the product implementation addressing another of the most important issues identified as one of the outcomes of the SLR, the difficulties to establish the relationship between the high level concepts on models definition and the product implementation.

This additional level is, thus, intended to provide the gateway between such Software Product’s observable attributes and the high level model’s characteristics so concerned with properties and operation factors such as software complexity, specification changing rate, operational environment or test coverage which, despite being of a lower abstraction level than the model characteristic, are still above the measurable attributes to with they are related.

The integration of this low level component allows us to obtain a complete description, from the user’s perception to the static measurable attribute and that mapping to the different user needs thus making explicit how the different stakeholder needs can be covered in an integrated schema. This simple layout is still a powerful description of the multiple facets of Software Reliability, in addition to providing the means to resolve the main issues extracted
from the literature review: the complexity of the descriptions and the difficulties involved in applying them as a decision tool in control and management activities.

The next figure 4.2 depicts our original layout for a structural model extended with these technological levels. It also show how it relates to the different reliability dimensions following the Garvin [50] proposal mentioned in Chapter 1. On the left side it also depicts how the model cope with the different stakeholders needs simply by construction.

![Figure 4.2: The Model in its context](image)

It is important to note that in Garvin terms, the transcendent view of Quality is somehow at the top of the hierarchy but being "a simple, unanalyzable property that we learn to recognize only through experience". The manufacturing and value-based views are related to Software Process, which is not the topic on this thesis, thus they are absent on the figure. Our interest in this research is on the product based and the user based dimension as well as on its relationship.

The product-based view, considers quality as a precise and measurable variable. According to this view, differences in quality reflect differences in the quantity of some ingredient or attribute possessed by a product. This approach lends a vertical or hierarchical dimension to quality in terms of such desired or possessed attributes. Quality differences could, therefore, be treated as differences in quantity, considerably simplifying the mathematics. The user-
based approach is an idiosyncratic and personal view of quality, and one that is highly subjective. Individual consumers are assumed to have different wants or needs, and those goods that best satisfy their preferences are those that they regard as having the highest quality. The new proposed model considers this relationship by construction, being the lower levels those capturing the product-based vision and the higher the user-based. The link is made by the model structure and operationalized by the appropriate aggregation techniques, in compliance which Standards proposals.

4.3. THE INDUSTRIAL APPLICATION FRAMEWORK

Before addressing the task of defining an application method, it is important to analyse the ecosystem in which such method will be carried out. Indeed, to apply the model in real-world situation it is required to develop a method or at least a methodology clearly stating the tasks to accomplish and the tools, mathematical or others that they will require. Unfortunately the International Standards are not of great help in this, thus we will develop these points in the next Chapter 5 where we will afford the definition of a method to apply this model. Also we need to verify the usefulness of the model, method and tools as a suitable framework for Software Product Reliability Analysis in daily Industrial Practice. We will also design an appropriate validation strategy on following Chapter 5.

As said, our purpose with this research is "To develop a Standard-Based Model for Software Product Reliability description and analysis as well as defining an application framework focused on its Industrial Applicability". Such Industrial Applicability is related not only to our model intrinsic attributes like soundness or accuracy but mainly with two essential extrinsic aspects; its ability to fit into the organization process with a minimum impact and minimal needs on resources (efficiency) and the capability to provide proper answer to user’s stated or implicit needs (effectiveness). In other words, it has to be profitable for the organization.

4.3.1. Profitable Software Engineering

Software Engineering is often treated as a mere technical challenge. However, in industry projects primarily have to ask what value a software engineering activity generates. Moreover,
[121] every industry those that produce the Software as those which exploit it are profit-oriented, then in absence of regulation Quality has historically been an add-on, of lesser market value than feature richness or short release cycles.

Although F.L. Bauer, who popularized the term "Software Engineering" at the NATO conference in 1968 already regarded the economic aspect of Software Engineering in his definition [91] "The establishment and use of sound engineering principles in order to obtain economical software that is reliable and works efficiently on real machines" such standpoint is not always as explicit. For example, for [57] Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. In [45] is said that Software Engineering activities include: Managing, Costing, Planning, Modeling, Analyzing, Specifying, Designing, Implementing, Testing and Maintaining of software.

Despite it should be clear that preventing defects is [125] the most cost-effective approach to Software Reliability. The reality is that it is extremely difficult to produce defect-free software and unrealistic to expect it in a fiercely competitive market. It is, thus required to make trade-offs between the delivered product quality and its costs. Bohem, who extensively considered economic aspects in software engineering [14] initialized the area of Value-Based Software Engineering. Among their insights is that to create as much value as possible during a software project, we have to be able to make economically balanced decisions. So how can we get such balanced decisions? [95] We have to make all effects of a decision transparent and we have to get all the stakeholders involved in the decision process in an adequate way. That implies the need for simple models and clear application methods accessible to the variety of stakeholders.

4.3.2. The Human Factor

It is our experience in industrial practice with a variety of large commercial systems that the particular nature of the Software Product which is not material but purely abstract production as well as its particular usage is also of paramount importance when considering the Software Industry. On the opposite to Hardware domain, Software Product Usage is usually neglected.
The authors of [121] note that usage is a central concern when testing hardware. The warranties for hardware items state that unintended use, which results in more than normal wear and tear, will render any warranties void. In [102], Jorge Romeu of the Reliability Analysis Centre of the (United States) Department of Defense mentions the success of the profiling for the validation of hardware and theorizes that this is because hardware components often work in a specific homogeneous environment. This homogeneity may be due to enforced standards. In contrast, the software operating environment is often extremely heterogeneous [102]. As it is explained e.g. in [121] where we can read how hardware warranties "often contain disclaimers about unintended usage" while Software is assumed being used in whatever way. Consequently, hardware reliability has seen more success than software reliability.

We need to carefully consider such reality providing means to deal with the "Human Factor" when developing the application method for our proposed model. That means that, among other points, we need to explicitly consider how the system is used by its actual users.

### 4.3.3. The Incremental Approach to problem solving

From the above, it is clear that we are obliged to work with economy of resources in an uncertain environment. This is true for both; our model validation as well as for its nominal exploitation. There are also several constraints within the process has to work; limited sources of information and limited availability of experts to complete the information when missed. We propose adopting an incremental process as best suited strategy in order to afford these issues. That because the use of incremental approaches have several benefits particularly relevant in the industrial practice.

Incremental approach offers to the project management a great control on the resources invested at the same time that a great flexibility in the development of the project. For these reasons, they are especially suitable for the resolution of industrial problems in uncertainty environments or when an analytical solution is not available at the moment project inception. By incrementally, we meant that a little more is added each time until the problem is solved. In that way it promotes the idea of minimizing risk and cost of the problem solving whilst building the necessary in-house skillset and momentum in an environment of high uncertainty.
Part II

Application and Empirical Validation
In empirical research it is common referring to the process of defining the measurement procedures for abstract concepts as "Operationalization". So it is Operationalization making our theoretical model applicable to existing Software Products in real-life industrial context. For that we need selecting a set of appropriate specific methods and procedures if they exist or to develop them if not existing. We also need clearly defining how they will be applied in practice pursuing the objective of having a straightforward to apply method. It is also requires to validate such application into real-world environment which is the ultimate ecosystem of any engineering production. We will afford such validation task by a Case Study application.

*Engineers turn dreams into reality.*
Hayao Miyazaki, The Wind Rises

5.1. INTRODUCTION

It is the very nature of engineering making real any solution theoretically derived or conceptually proposed to cope with real-life problems. This chapter is then, devoted to get an answer to the question: What do we need to do to apply our proposed model in compliance with the stated objectives? To get an answer we need to fulfil the following:

1. Overcome the conceptual complexity & variety of definitions

2. Capturing stakeholders expectations

3. Relate the product attributes to product behaviour
And moreover this is to be accomplished in a way which is understandable for the wider audience and straightforward to apply in order to promote its industrial application. It is from the answer to those points that we derive the validation objectives and the required operationalization procedures and techniques.

The model, by construction, deals with the conceptual complexity and by its simple structure promotes the ease of application. This simplicity also promotes making the analysis derived from it understandable to a variety of users, which is desirable to get the proposal accepted by non-expert stakeholders. The application method has to preserve all these. Still we need to capture stakeholder expectations as well as to relate the Product attributes to the Product behaviour. Along the following sections we will analyse how and why the Fuzzy Analytical Hierarchy Process and the International Standard recommendations can be applied as suitable means to deal with the human factor. In addition an explicit consideration of the Software Failure process and the integration of a specific realization of the Operational Profile concepts are also required to implement an appropriate framework to apply our proposed model. In practice, and for validation purposes, we will choose a specific group of users and a particular kind of software product attributes, the software defects since being the central factor on Software Reliability.

5.1.1. Aggregation Strategy

As mentioned one of the issues that hamper a broader application of the Standard Based approach is that the Standards provide little (if any) support to aggregate the low level measurements or assessments into higher level indexes capable to properly describe the understudy system’s characteristics. The challenge is to transform the raw data from lower levels in the hierarchy to produce condensed, or aggregate, information for decision makers. Too many indicators [94] compromise the legibility of the information which requires to be presented in a format suitable to the decision making. This requires the construction of indicators that reduce the number of parameters needed to give precise account of the general state of the system.

That lead us to the need of defining an aggregation strategy capable of soundly and efficiently sum-up such low level attributes accordingly to the model purpose, nature and the technological characteristics of the object under study. Such aggregation has to be
easily practicable and based on a well-founded measurement theory in order to ensure the validity of the analysis as well as to promote its application under industrial economic constraints. Moreover is also highly desirable that be ease to understand by the wider range of stakeholders.

5.1.2. Capturing Stakeholders view

Being one of our primary goals the development of models considering the user-based Reliability dimension we need a suitable way to capture the different Stakeholder’s views on system Reliability. Among the existing methods, the Analytic Hierarchy Process (AHP) [36, 107–109], is possibly the most well-known measure theory and a technique for the evaluation of complex concepts. This method is a well-suited option because its simple application and because in its Fuzzy implementation it is designed to manage human assessment, always characterized by a certain degree of vagueness and subjectivity. Although originally used in multiple-criteria decision making [109] AHP has a variety of applications [107] in economics, business, social sciences and engineering, and is widely applied in the fields of benefit analysis and risk assessment [107, 108, 114]. Considering that risks assessment is analogue task to Quality assessment we preconize the suitability of this approach. On the following we will show how this method can cope with both, stakeholders’ assessment and views and the aggregation task.

5.1.3. Failure Mechanism

Another of the challenges, arguably the main, when analysing Software Product Quality is making the link between the high level Quality characteristic and the low level source code attributes whatever are required features or undesired defects, which are the ones of most interest in the particular case of Reliability analysis. This is also needed to operationalize the structural model in a way that can be applied to software product assessment establishing such relationship between software code defects or “bugs” and Reliability attributes consequence of the exhibited behaviour. We need, then, to analyse the software failure mechanism and for that the way the system is used when in exploitation environment. We will apply the common understanding on the failure process since it is the mechanism that leads from a defect in the product to a malfunction of the system. We mainly followed Lyu [77] and IEEE [55] in order to fix a terminology and to describe the failure process in a suitable way
to our purposes.

The agreement on the failure mechanism is not enough by itself, for a proper systematic analysis we also need to classify our identified defects, faults and failures in a way that will allow us to derive sound conclusion on its relationships, and then take the corrective or mitigation actions. We need thus a suitable taxonomy for Reliability-related software product defects. Unfortunately such a taxonomy fitted to our purposes does not exist. Actually no commonly accepted taxonomy exists for software defects instead a variety of proposal more or less well suited for particular scopes and point of views have been proposed in the past. We face, thus the need of producing an appropriate one. As a starting point we chose doing that on the basis of existing bugs taxonomies in the aim to produce our own revised Software Defect Taxonomy to apply as a reference tool in our further analysis. We will develop that problem along the next section 5.2.

5.1.4. Software Usage Modelling

Closely related to the failure mechanism is the way the system is used since largely determining whether the failure happens or not. We need then to integrate a technique that will allow us to take account on that circumstance as a modulator of the failure process. The standard proposal in Software Engineering is the Operational Profile (OP) in probabilistic terms. Notwithstanding developing this kind of OP is difficult, expensive [121] and has been criticised as easily lacking of accuracy. None of that goes in the sense of promoting industrial applicability. As a consequence we will afford the definition of the Operational Profile in a non-probabilistic manner, straightforward to obtain from;

1. The expert’s knowledge on the system.
2. The business definition and mission.
3. The high level system operation logs.

Still in an effort to promote the industrial application by low impact straightforward to apply procedures. This also means that our approach is not just a black-box but that knowledge on the structure, functionality and the way the system is actually being exploited is integrated within it.
5.1.5. The Case Study

Finally, we need to validate that our proposed model and application method are actually feasible and performable in real world, thus within common industrial constraints. As thoroughly developed in Chapter 2 among the methods of the Empirical Software Engineering, Case Studies reveals as the better suited approach since developed as "an empirical method aimed at investigating contemporary phenomena in their context" while exploiting "multiple sources of evidence" within the constraints that impose a "lack of experimental control" which is exactly the environment we face when proving our model and framework in real-world industrial conditions.

Section 5.5 will present the design of the Validation Process for the proposed Model & Framework as an Incremental Case Study strategy developed in this thesis to allow their empirical validation. This is one of the contributions of this Thesis. A particular implementation of the Case Study methodology in a new method oriented to afford the problem analysis from the simpler approach to the whole realization by means of steps of increasing scope. The overall objective of the Model Validation is to analyse and prove its Industrial applicability and usefulness as a valuable industrial tool. That means proving the soundness, feasibility, descriptiveness and Industrial Applicability in terms of efficacy and effectiveness.

5.2. THE AGGREGATION STRATEGIES

In order to minimize the amount of data and made understandable the proposal we need, as it is common practice, making use of aggregate indices. The objective of such aggregation being [94] assist decision makers by reducing the clutter of too much information, thereby helping to communicate information succinctly and efficiently. Ultimately, when making a decision the decision maker must go through a process of condensing information to make simple comparisons. An aggregation function formalises what is often done implicitly. Critics of aggregate indices point to the loss of information and the unrealistic assumptions made to formalize the aggregation as potentially introducing serious distortions that can lead the observer to misinterpret the data. An example is the "compensation effect" when aggregating by means of more or less sophisticated average computation. Also, as Littlewood explain in [76] the statistical methods have intrinsic limitations, when claiming [19] that "MTBF is
meaningless in software reliability” or when analyzing Software Reliability Grow Models [75] states that "it seems clear that such methods are really suitable for the assurance of relatively modest reliability goals". In general, all these issues have the same common intellectual origin which is the consideration of Software Reliability as similar to the Hardware Reliability despite the fundamental dissimilarities between them.

A better option for us is found on the Fuzzy implementation of the Analytic Hierarchy Process since synthetizing the ability of fuzzy techniques to deal with the characteristic vagueness of human perceptions and assessment as well as the proven capabilities of AHP to manage problems where multiple concurrent criteria have to be taken into account as such is the case in our problem. In that way we can tackle the practical issues related to the stakeholders’ point of views and variety of needs and operationalize the complexity of the Reliability concept.

**Fuzzy Analytic Hierarchy Process:** The Fuzzy Analytic Hierarchy Process (FAHP) is a generalization of the classic Analytic Hierarchy Process (AHP) in the aim to deal with the impreciseness and vagueness of human assessments.

The Analytic Hierarchy Process was proposed in 1970s by T. L. Saaty [108, 109], an American expert in Operational Research. AHP [108] is a mathematical theory for measurement and decision making. It was developed [108] as a reaction to the finding that there is a lack of common, easily understood, and easy-to-implement methodology to enable the making of complex decisions. AHP was not developed searching for an optimal solution but [107] rather advance towards an acceptable solution, which is quite the nature of real-world engineering practice. It is worth to note that the decision-making process always involve an assessment task on complex concept or systems. Such assessment task is which is of more interest in our analysis.

**5.2.1. The F-AHP fundamentals**

This approach main feature is the hierarchical decomposition of the assessment problem. Once the hierarchy is built, the domain experts systematically evaluate its various elements, comparing them to one another in pairs. In making the comparisons, these experts can use concrete data about the elements, or they can use their judgments about the elements’
AHP offers a number of advantages that makes it suitable for our purposes. The main advantages are handling multiple criteria, easy to understand and effectively dealing with both qualitative and quantitative data. Its appropriateness to our purposes is also supported by the fact that AHP is a sound and well-founded mathematical theory. A notable strength is its ability to detect inconsistent assessments. Also AHP uses of Hierarchy Decomposition as fundamental problem solving technique which a well-known and broadly applied approach, e.g. it is a common approach to many International Standards on Software Quality, notably SQuaRe as well as its predecessors ISO/IEC 9126 [62], Mc-Call [84], Boehm [15] and many others.

These characteristics make AHP an effective methodology to a wide range of applications and allow any complex situation that requires structuring, measuring, and synthesizing to be better understood. Like all modelling methods the AHP has also weaknesses, it is known that presents limits to capture the characteristic vagueness of human thinking. In order to overcome this limitations the Fuzzy AHP approach was introduced by Laarhoven and Pedrycz [72] as a merger of the classic AHP and Fuzzy logic concepts [128, 129] and tools to gain advantage of both and, therefore, addresses the above mentioned problems.

The Fuzzy-AHP method has been widely used by various researchers to solve different decision making problems. We have ourselves made a first application of Fuzzy AHP to Software Product Reliability [42]. On this thesis a more elaborated and extended application expands this way.

The FAHP, as the AHP, is developed along three main phases:

**First**, the problem has to be modelled into a Hierarchy which indicates a relationship between elements of one level with those of the level immediately below. The obtained model is thus, a structural one.

Once the hierarchical model has been built two kind of data need to be collected from domain experts. Relative comparisons and performance assessments. The objective of these relative comparisons is comparing the hierarchy elements to one another with respect to
the impact on their parent node. This is achieved by means of the expert panel’s answers to questions of the general form, "How relevant is element A as regards to element B?" known as "pairwise comparison". Answer to the question can be done in a variety of expression but the more natural is in the form of what in this context is called, Linguistic Variables; Equally important, Moderately more important, strongly more important, very strongly more important, extremely more important. Many other similar expression are also valid.

Then, these pairwise comparisons gathered at previous step are now arranged into the comparison matrix at each level and according to the hierarchy structure. A comparison matrix is a square matrix $A = [a_{ij}]$ where the element $a_{ij}$ is the relative importance of element i compared to element j. The diagonal elements of the matrix are 1 and $a_{ji}$ can be assumed to be $1/a_{ij}$.

Now, the vector representing the relative weights, or local priorities, of each of the attributes, can be found by computing the normalized eigenvector corresponding to the maximum eigenvalue of each matrix. This vector can be computed by dividing each element of the matrix with the sum of its column, then averaging the values in each row.

Finally, to aggregate the priorities along the hierarchy we have to consider that local priorities are calculated among same level elements and global ones are the product of all local weights from leaf to root on the hierarchy. In other words, the local priorities correspond to the relative weights of the nodes within a group of siblings with respect to their parent. Global priorities are obtained by multiplying the local priorities of the siblings by their parent’s global assessment.

Once the model built and parameterized in such way it can be exploited. To do so a second kind of data, the performance assessment, is required.

### 5.2.2. How to apply F-AHP In practice

There are several FAHP methods reported in the literature, main ones are those from by Laarhoven and Pedrycz [72] Buckley [20] and Chang [29]. In this work we will make a rather simple application since our objective is not on the Fuzzy AHP technique itself but on the model validation. On the following we will detail, step by step, our method for FAHP
application in this research.

**Step 1: Set-up the problem and evaluation criteria** Problem has been modelled in a hierarchy, what we have done with our proposed Standard Based Model as per figure 4.2 in Section 4.2, which is fundamental to the process of the AHP. The assessment criteria are defined and explained to the participants. In this particular case such agreement is about the actual meaning of the SQuaRe definitions and Linguistic Variables.

**Step 2: Gather each participant assessment** The expert panel is asked to make their assessment. There are two different assessments; the one regarding the pair-wise comparisons of the importance between each pair of sub-characteristics in the SB-SRM model. On the other hand stakeholders can also be requested to provide their evaluation on how the System under scrutiny deliver is intended functionality in terms of performance, behaviour, completeness, etc.

Decision-makers and domain experts usually feel more confident [97] to give linguistic variables, rather than expressing their judgments in the form of numeric values. Hence, the fuzzy set theory is a useful tool in dealing with imprecise and uncertain data. The assessment is, therefore, in the form of previously agreed linguistic variables. Depending on the assessment purpose or compared elements the appropriate Linguistic Variables will be different.

**Step 3: Operationalize the assessment** Then the FAHP method implies converting the linguistic variable to fuzzy numbers [129] to which apply the computations.

In this work, without loss of generality, Triangular Fuzzy Numbers (TFN) will be used to represent those subjective comparisons. For doing so a scale definition as the one proposed by Chen [30], is required to convert such linguistic variables into TFNs. Literature provide a variety of scale mappings. It is not our aim discussing the goodness of one or another. In this work our choice will be driven by simplicity. They are shown in table 5.1. Since the application of the FAHP algorithm, as we will see on the following requires the use of reciprocal numbers (when computing the comparison matrix) they are, by convenience, also detailed in the table 5.1.
### Linguistic variable | Triangular Fuzzy | Reciprocal Triangular
--- | --- | ---
Equally important | 1,1,1 | 1,1,1
Intermediate values | 1,2,3 | 1/3,1/2,1
Moderately important | 2,3,4 | 1/4,1/3,1/2
Intermediate values | 3,4,5 | 1/5,1/4,1/3
Essentially important | 4,5,6 | 1/6,1/5,1/4
Intermediate values | 5,6,7 | 1/7,1/6,1/5
Very strong importance | 6,7,8 | 1/8,1/7,1/6
Intermediate values | 7,8,9 | 1/9,1/8,1/7
Absolutely important | 8,9,10 | 1/10,1/9,1/8

**Table 5.1: Fuzzy Assessment**

#### Step 4: Consistency Check of Individual Assessments

Those provided data need to be reviewed in order to ensure the internal consistency of each of the provided set of pairwise comparison. Inconsistent data will be rejected. T.L. Saaty [108] proved that for consistent reciprocal matrix, the largest eigenvalue $\lambda_{\text{max}}$ is equal to the size of the matrix, $\lambda_{\text{max}} = n$. Then he gave a measure of consistency, called Consistency Index as deviation or degree of consistency using the following formula:

$$CI = (\lambda_{\text{max}} - n)/(n - 1) \quad (5.1)$$

This largest or principal eigenvalue is obtained from the addition of products between each element of the eigenvector and the sum of columns of the reciprocal matrix.

In this simple approach, each triangular number is transformed to a non-fuzzy or crisp number, we choose the modal value just for simplify the computation. Then, the eigenvector is obtained by dividing each element by the sum of its column then adding by rows and normalizing by the matrix size. Finally, the consistency ratio (CR) is defined as a ratio between the consistency of a given evaluation matrix and consistency of a random matrix:

$$CR = CI/RI(n) \quad (5.2)$$

Where $RI(n)$ is [36, 107] a random index that depends on $n$, as shown in Table 5.2.

Those assessments that lead to $CR \leq 0.1$ are consistent enough as to be considered in
the analysis, thus will enter in the next step, which is the aggregation of the individual assessments into a unique set named "consensus". Those which do not satisfy that condition have to be rejected.

**Step 5: Synthesis of individual assessment into consensual values** For the aggregation of every participant assessment into a single consensual judgement matrix there is a variety of proposal available in the literature. We will apply the simple geometric mean method.

In summary this approach consists in, for triangular fuzzy numbers in the form \((l_i, m_i, u_i)\), choosing the minimal value among the lower elements \(l_i\), the maximal among the \(u_i\) upper elements and the geometric mean of the \(m_i\) modal values. Taking into consideration the layout of the under analysis hierarchy we will typically obtain a set of matrix which sum-up the stakeholders assessment of the under analysis system.

**Model Parameterization** There are several methods to compute the fuzzy final weights representing the global assessment on relative importance of the structural components of the model. It consists in an extension of Saaty’s procedure to fuzzy reciprocal matrices, and was first introduced by van Laarhoven and Pedricz [72]. Also Buckley [20] method is frequently cited although considered of complex application. The more popular seems to be the extent method by Chang [29] which is of ease application but prone to be mistaken.

In our present application we choose, for the sake of simplicity, an approach based in the geometric mean.

1. We can determine the model parameters by using geometric mean technique [20]. First, geometric mean is computed by rows and then it is normalised.

2. Then those triangular numbers are translated onto crisp or "de-fuzzified" values, for doing so there is a variety of means reported in the literature [16], we will apply the simple centroid method. Then, those crisp values are normalized.
3. Finally we obtain the complete structure of relative importance of each element in
our structural model accordingly to stakeholder point of view. Those are the model
parameters.

**Model Application** The performance assessment on the exhibited performance for the
same attributes for which they gave the relevance judgment is processed in a similar manner.
Linguistic variable are translated into TFNs from which a consensual assessment is computed
by geometric mean and centroid method.

### 5.3. FROM DEFECTS TO FAILURES

Commonly speaking the terms defect and fault are used interchangeably additionally to the
term "bug" in Software Engineering context. They are often defined differently by different
sources including the imperfections found in the code during technical reviews [81]. Also
fault, failure and breakdown are often confused. Such vagueness in terminology is an issue
that need to be overcome in order to produce a proper analysis.

#### 5.3.1. The Software Failure Mechanism

A suitable starting point can be found in the IEEE Standard Glossary of Engineering Ter-
minology [57] where it is stated that a FAULT in a software artefact is an incorrect step,
process or data definition. Under suitable circumstances a fault may cause a FAILURE which
is understood as "the inability of a system or component to perform its required functions
within specified performance requirements" that is, a deviation from the stated or implied
requirements. Besides software faults and failures may also be caused by improper input
either by human actor or some component of the system and by failing hardware or other
equipment.

A step forward is considering that at the origin of the software failure there is always a
human error. Following Wohlin an error or mistake is then a human action, commission
or omission, like forgetting a particular case, the misinterpretation of some information,
an incoherent specification, etc. resulting in the software containing a DEFECT which is a
wrong or undesired feature or attribute. In summary, an error is the action that creates a
defect. Such defect can remain latent in the code for an undefined time. Only in the case of
some specific execution conditions are met such defect will result in an execution FAULT of
the functional unit related to the program where the defect has manifested, which is not
necessarily the very same carrying the defect. Finally, depending on the nature of the fault
it will or will not lead to a software FAILURE which ultimately could, or could not, end up
in a system breakdown. Wording all that as formal definitions we propose the following;

**Error:** An error is the human action that results into a defect in the software product
documents or implemented source code - it can be an omission or misinterpretation of
user requirements in the software specifications, incorrect translation, or omission of a
requirement in the design document

**Defect:** Is then an undesired or wrong attribute or the lack of a required right feature or
attribute detectable from source code or the design or operation documentation. In SE context
it is usual referring to them as "bugs" when in the source code. Defects are consequence
of human errors. Examples include; missing a required check, wrong initialization, data
reference out of bounds or misleading instructions in a software operation document.

**Fault:** A fault is an accidental condition that causes a particular piece of code or a functional
unit to fail to perform its required operation. Faults are the consequence of defects under
specific execution conditions. Examples include; memory leaks or data corruption, e.g.

**Failure:** A failure is an event where a system or a component of the system stops performing
a required function, ceases to deliver the expected service within acceptable boundaries or
the user perceives that a software program behaves in deviation from its expected course
during execution. In other words, a software failure is an incorrect result with to the
specification or unexpected software behaviour perceived by the user. Examples include;
process crash, system hung.

Failure is the observed impact of faults. Often, but not always, the fault leads to a failure
thus acting as the underlying cause of the failure.

In other words, for a defect resulting in a failure in any way observable by the end user it
is required [19] not only that the defect to exists in the system but also that the piece of
code carrying it being executed. Moreover the execution conditions are those that lead to a
5.3. FROM DEFECTS TO FAILURES

Figure 5.1: The Failure Mechanism

wrong condition or output which, depending on the specific operation situation makes the system to exhibit a behavior in any way undesired by the user. Let say that the first makes the "Execution Context" and the second can be named "Operational Context". Such is the fundamental reason why the way the system is used is as relevant as the software product itself in terms of Reliability. This leads us to the need of modeling the system usage as we will see on next section 5.4 but before we needs the means to analyze the software defects itself.

5.3.2. Defects Taxonomies.

An explicit taxonomy is necessary to support the systematic analysis of the defects and its consequences on the system behaviour. It also promotes coherence and reproducibility of such analysis.

We have just seen that there seems to be no universally accepted definition of the term defect, as a consequence it is not surprising that the Software Product defects classification is also an open issue with a variety of proposal and different approaches. It is not our aim to investigate here the Software Product Defects classification problem but in a limited extent. Our objective is just to get an appropriate response to our practical need which is deriving a Reliability related classification of software product defects.

Notwithstanding this classification schema do not purport being comprehensive and of general application for any possible defect but it aims being coherent, consistent and complete enough as per our intentions in relation with product reliability for what we are mainly
interested on the defects relationship with software quality attributes, (which is less studied and makes part of our CS outcomes). In other words, we are only interested on those product defects related to system behaviour. We do not consider security related flaws or maintainability problems e.g. lack of appropriate embedded comments in source code which, despite being bad praxis have no incidence on system behaviour.

In that sense and because we are interested on technical nature of Software Product defects, flaws on testing design or execution although having an impact on the final Quality of the delivered product are out of scope of the present analysis. In the same way, some coding bugs like wrong comment on code having no impact on system behavior when on exploitation are also kept aside as well others in the same vein.

It is then, on this specific purpose that we have identified and have critically analysed the, to our understanding, more relevant previous contributions to the topic, table 5.4, identifying their common points and shortcomings to construct our defect taxonomy based on these prior works and our own data as we will develop along the present section.

We have followed a quasi-systematic analysis procedure on four stages; previous contribution selection, overview and context understanding, defect types conceptual clustering and final synthesis with a particular attention to completeness and unambiguousness of the proposed definitions.

**Previous contribution selection**

We will proceed in a similar to the formal Systematic Literature Review starting with the following bibliographic query.

<table>
<thead>
<tr>
<th>Source</th>
<th>IEEE-Digital Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>((defect OR error) AND (taxonomy OR classification)) AND software)</td>
</tr>
<tr>
<td>Filters Applied:</td>
<td>IEEE Journals &amp; Magazines Early Access Articles 2009 - 2018</td>
</tr>
</tbody>
</table>

**Table 5.3: Search On Software Defects Taxonomies**

The reduced number of results allowed explicit review. As it was the case during the SLR most of the hints are discarded by reason of not Software Engineering Topic, not Software Product, etc. The few which are potentially in-scope are processed by analysing title, abstract and keywords. That leaded to locate two previous contributions [81, 105]. In order to locate
more relevant works we applied snow-balling technique on references starting on these ones. In that way we got to select the following nine references:

<table>
<thead>
<tr>
<th>Ref</th>
<th>Author</th>
<th>Title</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>[7]</td>
<td>Boehm</td>
<td>Some experience with automated aids to the design of large-scale reliable software</td>
<td>1975</td>
</tr>
<tr>
<td>[32]</td>
<td>Chillarege</td>
<td>Orthogonal Defect Classification A Concept for In-Process Measurements</td>
<td>1992</td>
</tr>
<tr>
<td>[105]</td>
<td>Runeson</td>
<td>What Do We know about defect detection Methods?</td>
<td>2006</td>
</tr>
<tr>
<td>[81]</td>
<td>Mantyla</td>
<td>What types of defects are really discovered in code reviews?</td>
<td>2009</td>
</tr>
</tbody>
</table>

**Table 5.4: Selected Previous Software Defects Taxonomies**

**Overview and context understanding**

At the higher level Runeson et al. [105], defects are often classified according to three dimensions: First, a defect can be classified as either an omission or a commission, second, a defect may be classified based on its technical content and third, defects may be classified by the impact on the user.

We are more interested on the second dimension, its technical nature and how that relates to the third one, the impact on the user. This aim implies investigating means to characterize the technical dimension in a way that will allow inferring from there the exhibited behavior and thus the impact on the user. This is challenging since regarding its technical nature software defects are recognized difficult to categorize. For example, a one-character error in a source statement changes the statement, but unfortunately it passes syntax checking. As a result, data are corrupted in an area far removed from the actual bug. That in turn leads to an improperly executed function. Is this a typewriting error, a coding error, a data error, or a functional error?

Among the selected taxonomies we distinguish two groups, on the first one the proposal from Beizer and that from Kaner are developed to a very detailed low level source code attributes while on the second, the rest of taxonomies stops the classification at much more high level categories. On the first group, Beizer and Kaner proposals, the detail level looks a priory
too demanding in resources and staff training as to be of great benefit in industrial context unless having ad hoc tools to greatly support such classification and further processing. On the second group, the detail level looks insufficient to inform by itself on a relationship between defect type and its impact in the Software Reliability sub-characteristics.

Another issue is that the description level on those available taxonomies do not fit our purposes. Either it is of high level thus insufficient to inform by itself on a relationship between defect type and its impact in the Software Reliability sub-characteristics, either the description level is so low level detailed that the classification task is too demanding in resources and skills. This is a clear impairment to be of ease application in an industrial context which is one of our main objectives. It is our aim to deem in which extent such level of detail is of value or constitutes a flood of data unsuited for our purposes.

All those previous taxonomies are oriented to identify the relationship between software defects and its origin and causes in order to inform the software process in the aim to develop flawless product, better debugging techniques and more efficient testing. In other words are process-oriented. For us, on the contrary, the goal is about to correlate the defects with their impact in terms of product behaviour, let say that we seek for a product-oriented bugs taxonomy. Related to this preponderance of development process is also the fact that Recovery related issues which are arguable strongly related to operation procedures are not explicitly treated and looks sunk in the generic consideration of documentation issues.

In addition there is some confusion in many of them on whether they cope with defects, faults or failures which is mayor impairment to its application for our stated purposes. In the same vein, all these previously proposed taxonomies suffer of the same problems of ambiguousness on the definitions and heterogeneity in the description level which often do not attain the source code level or not in a consistent manner when it does. Also, is general that concepts definition are missing and thus proposed classifications are prone to misconceptions.

**Defining a new Software Defects Taxonomy** As a consequence of all the above we have developed an ad hoc taxonomy oriented to our declared specific goals. This taxonomy takes up on the categories that made a consensus along the reviewed previous ones and is amended and completed with categories of our own proposal for those which are missed, poorly
defined or which shows no agreement among the previous contributions. The conceptual clustering has given us a first level which need to be developed to Software Attribute level, thus software defects as we have defined on section 5.3.1

On the following table 5.2 we synthetize the findings from its analysis. Further details and definitions are to be found on Appendix B.

<table>
<thead>
<tr>
<th>Software Defects Taxonomy for Reliability Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Functionality</strong></td>
</tr>
<tr>
<td>1. Missed feature or requirement</td>
</tr>
<tr>
<td>2. Missed Function</td>
</tr>
<tr>
<td>3. Wrong feature or requirement</td>
</tr>
<tr>
<td>4. Wrong function</td>
</tr>
<tr>
<td>5. Missed case</td>
</tr>
<tr>
<td>6. Missed check</td>
</tr>
<tr>
<td>7. Wrong case</td>
</tr>
<tr>
<td>8. Wrong check</td>
</tr>
<tr>
<td><strong>Definition</strong></td>
</tr>
<tr>
<td>9. Wrong data structure</td>
</tr>
<tr>
<td>10. Wrong database schema</td>
</tr>
<tr>
<td>11. Wrongly dimensioned data</td>
</tr>
<tr>
<td>12. Contents misinterpretation</td>
</tr>
<tr>
<td><strong>Handling</strong></td>
</tr>
<tr>
<td>13. Incorrect data conversion</td>
</tr>
<tr>
<td>14. Inconsistent data population</td>
</tr>
<tr>
<td>15. Data reference out of bounds</td>
</tr>
<tr>
<td>16. Data corruption / integrity</td>
</tr>
<tr>
<td>17. Extraneous output data</td>
</tr>
<tr>
<td><strong>Coding</strong></td>
</tr>
<tr>
<td>18. Wrong logic or algorithm</td>
</tr>
<tr>
<td>19. Path left out</td>
</tr>
<tr>
<td>20. Dead end code</td>
</tr>
<tr>
<td>21. Improper nesting</td>
</tr>
<tr>
<td>22. Duplicate processing / Unnecessary processing</td>
</tr>
<tr>
<td><strong>Processing</strong></td>
</tr>
<tr>
<td>23. Wrong initialization</td>
</tr>
<tr>
<td>24. Arithmetic</td>
</tr>
<tr>
<td>25. String wrongly handled</td>
</tr>
<tr>
<td>26. Wrong format</td>
</tr>
<tr>
<td>27. Exception or error not handled</td>
</tr>
<tr>
<td>28. Boundary condition</td>
</tr>
<tr>
<td>29. Typo and clerical error</td>
</tr>
<tr>
<td><strong>Integration</strong></td>
</tr>
<tr>
<td>30. Invalid input</td>
</tr>
<tr>
<td>31. Wrong library</td>
</tr>
<tr>
<td>32. Lack of mutex</td>
</tr>
<tr>
<td>33. Wrong mutex</td>
</tr>
<tr>
<td>34. Wrong message or parameters</td>
</tr>
<tr>
<td>35. Wrong or un-existing routine called</td>
</tr>
<tr>
<td><strong>External Interfaces</strong></td>
</tr>
<tr>
<td>36. Wrong I/O protocol</td>
</tr>
<tr>
<td>37. Assumptions on external I/O</td>
</tr>
<tr>
<td>38. Wrong resource used</td>
</tr>
<tr>
<td>39. Resource deadlock</td>
</tr>
<tr>
<td>40. Device error handling</td>
</tr>
<tr>
<td><strong>Documentation</strong></td>
</tr>
<tr>
<td>41. Lack of recovery procedure</td>
</tr>
<tr>
<td>42. Wrong recovery procedure</td>
</tr>
<tr>
<td>43. Wrong operation procedure</td>
</tr>
</tbody>
</table>

**Figure 5.2: Reference Defects Taxonomy**

### 5.4. MODELLING THE SYSTEM USAGE

The ultimate goal is on modeling how the defects in the product impact on the system behavior. As discussed before, for a defect resulting in a failure in any way observable by the end user it is required [76] not only that the defect exists in the system but also that the piece
of code carrying it being executed. Such is the fundamental reason why the way the system is used is as relevant as the software product itself in terms of exhibited behavior. That means the need to model how a defect manifests itself or not in a failure and the relevance of such failure in the perception of the "degree to which a system, product or component performs specified functions" when exposed to the user.

The importance of considering such mechanism is recognized since the inception of the topic of Software Reliability, Littlewood as early as 1977 [19] when analyzing uncertainties on input space to programs and again in [76] when says "a program with two bugs in little exercised portions of code can be more reliable than a program with only one, frequently encountered, bug". In such sense, Musa [86] states that a software-based product’s reliability depends on just how a customer will use it, thus proposed the use of models built as [86] "a quantitative representation of how a system will be used" and coined the term "Operational Profile" to refers to.

5.4.1. The Operational Profile

And Operational Profile is then, a quantitative model of how the system is used which is intended to be expressed in terms of probabilities of a particular piece of code to be used during the nominal operation of the system as "the set of operations or processes for a software application, and the probabilities of occurrence for those operations or processes" [86], as a way to describe how actual users operate the system.

This system modeling usage was initially introduced in the S.E. body of knowledge [88] in the testing area recognizing that a close match with actual usage is necessary otherwise the analysis outcomes will not be reflected in the actual usage of the system.

Using operational profile as the test basis is recognized as a good approach to rapidly increase reliability by placing the software of interest in a testing circumstance that is typical of the circumstance in which it is expected to operate [86]. Although it was developed to guide testing, the operational profile can also guide managerial and engineering decisions throughout the life cycle. Musa [87, 89] suggest the operational profile can also be used in performance analysis. The operational profile also improves communication between different stakeholders and within the organization by making expressions of needs more
Notwithstanding there are several drawbacks to consider. Although Operational Profiles are recognized useful to afford a variety of problems in Software Engineering, there are important challenges; conceptual and practical ones that hamper to properly apply such Operational Profiles as a reliable system model either on research or industrial environments.

Building-up an accurate probabilistic OP is difficult often very demanding on resources and also impairments often exists linked to the availability or cost of getting the required data. Moreover, the distortions that instrumenting the system to obtain these data can introduce greatly hamper or even avoid the attainment of a practicable model. Indeed, this classic concept based of "occurrence probability" is not always suited or even impossible to obtain because has the underneath presumption that every possible action is evenly happening and important. We can also add that probability concepts are often mistaken by non-experts which results in difficulties to interpret the results and the benefits of its usage within the Software Life Cycle.

The fact that the piece of code carrying the defect has to be executed to the misbehaviour exists implies an associated operation context which is also relevant for the perception of the exhibited behaviour as an issue. In other words, the same defect can result in a variety of misbehaviour with different consequences in system, circumstance which is not captured by the classical probabilistic approach.

None of that goes in the sense of promoting the classic probabilistic approach as an industrial tool. Also our purpose differs since we do not pursue driving testing activities but to integrate the system usage modes as a modulation function into the Reliability assessment process. Other authors [121] has already point the original definition in probabilistic terms as too narrow and proposed extensions and modifications to make the concept applicable to other purposes than testing the software.

In this study we look then for a lighter but also useful approach.
5.4.2. Our proposed alternative

Because of all the above our approach to model the system usage will be different to the classic one. In a more general view an Usage Model is a qualitative or quantitative characterization of how a system is used. Following this vision we will consider a non-probabilistic characterization of how the system is used, although still considering usage frequencies as a valuable input among others but not the central one.

We have, then, developed a usage model understood as an analytic description of the system exploitation patterns, considering when, how and why each of the system facilities is required. We do that on the basis of considering the Business model, that is the mission of the under study system, its functional areas which constitutes the technological expression of such mission as well as the critically and the temporal characteristics of its utilization.

In practice, to determine the usage profile, we will look at the system use from a progressively narrowing perspective, from general business goals to system end operator ones. At each step, we will assess how often each of the functional elements in that step will be used and how relevant are for the mission to achieve. Functional areas are defined from the users’ perspective and do not involve architectural or design factors. We will also integrate the criticality of the different functional areas since not all are evenly important for the mission.

**Step 1: Business definition and mission goals** Relevance of failures is bound to business mission the software cope with. It is then the starting point to clearly stating such business needs or appointing a business expert to the task of produce the usage model.

**Step 2: Mission Operations identified and enumerated.** In order to achieve the business goals a variety of operations need to be successfully achieved. Are those operations which the Software Product is expected to implement which are of interest for the Usage Model definition task.

**Step 3: Functional splitting** The implementation of such required operations by means of a software product requires of a functional description at a suitable decomposition level.

**Sub-step 3.1. Time constraint assessment** In any mission will find that the required operations are under different requirements in terms of when have to be successfully
accomplished, some will need to succeed in very precise moments other could be just performed at some moment, this need to be captured in the usage model since defining how relevant each particular failure will be.

**Sub-step 3.2. Domain constraint assessment** In any system will find that among the variety of available facilities to accomplish the mission some are of utmost importance for the business goals while others are on the category of nice-to-have, this need to be captured in the usage model since defining how relevant each particular failure will be.

Both assessments can be easily produced by domain experts using Linguistic Assessment as we have developed in previous section 5.2.

**Step 4: Relevance Computation** For each functional area under consideration we will compute its associated relevance as a function of its domain and time constraints. In our approach we will use standard fuzzy numbers arithmetic although this process admits any other way of computation of the joint effect.

**Step 5: Usage Relevance Model** Finally we will get as outcome a set of functional areas together with its associated relevance to achieve the system intended mission. This model is explicitly abstracting not only the system usage but also the user’s standpoint on it.

This is a top-down process which will stop when judged the attained description level is suited to the analysis objectives. The sources of information are not limited to the domain expert knowledge the project documentation when existing is a valuable input on which to get support for the process, only when the product is not properly documented we will limit to rely on expert’s judgement. The frequency of execution dimension will be captured later when processing each individual failure event which is much more precise and affordable since having available much more detailed information on each defect.

**5.5. THE INCREMENTAL CASE STUDY STRATEGY**

There is no yet standard or universally agreed application method for the Case Study methodology for Software Engineering, instead and on the basis of the commonly accepted tasks to perform in Case Study each particular application is being designed in a case by case
manner. Much of the up-today applications are of a one-shot waterfall sequential application of some or all of the typical Case Study tasks that we presented in Chapter 2; definition, data gathering, application, analysis and report. In [123] a short of Waterfall-like application is reported and in [104] there is an example of iterative application over several cycles till meet some specific predefined criteria.

As we have discussed in previous Chapter 4 the use of incremental approaches have several benefits particularly relevant in the industrial practice. Among them that of providing high control over the involved resources as well as being well suited to tackling situations of high uncertainty as it is this one we afford here. As a consequence in our application we have developed an alternative to those aforementioned which, at the best of our knowledge, is a novel approach; the Incremental Case Study strategy. We define this Incremental Case Study (I-CS) process in order to enable stepwise realization of the general goal from the simpler scope to more complete and complex realization.

In our application strategy the case definition, the involved stakeholders, objectives and data are defined and gathered as a part of the initial steps of the protocol as it is done in the classical waterfall-like applications but not in the iterative way as shown in [104] where each iteration implies that they are reassessed. Then, and differently to the classical approaches the application is done, not to the whole under study object but partially from the more simple in terms of both structure and objectives to, by means of steps of larger scope, reach the whole picture. Each of these increasingly complete and complex steps will cope with particular objectives, which all together constitutes the model validation as well as the application of our proposed model in a framework that allows minimizing implementation risk and easy adaptation to different stakeholder levels.

There are, then, fundamental differences between this new approach and the more applied waterfall or the mentioned iterative strategies; these can be seen on the following table 5.5 which presents a comparative of the implementation of the main Case Study tasks in each of the three application methods.

The first simple stage is mainly an exploratory realization seeking for confirmation or refutation of our basic postulates in a minimal risk set-up. The final case study stage means an explanatory approach with the purpose of finding, in some extent, causal relationship
that could explain observations and predict future behaviour.

In our application strategy, as can be seen on the table 5.6, we have a first step focused on feasibility and conceptual confirmation for which we choose a simple implementation of Decision Making related techniques. We consider this as suitable approach because at this high level are the managerial consideration which prevails in the analysis. Our second application is a real world industrial application intended to capture stakeholders’ perception for which a whole realization of a FAHP is an efficient tool as we will detail on the following. Finally the third incremental step being focused in the integration of the product attributes impact on the system perceived behaviour will need of a way to consider how the failure process drive this. We will develop for that an ad hoc treatment, the Usage Model, starting on the well know concept of Operational Profiles although developing a light straightforward to apply novel approach intended to promote applicability.

In that way our realization validates, first the feasibility of the approach then, as per the schema presented in figure 4.1, its ability to capture the Stakeholder’s expectations on Software Product Reliability and finally will afford the more complex problem, as depicted in figure 4.2, of the relationship between the low level attributes and the exhibited behaviour in order to understand the extent to which the proposed Model and Framework is able to overcome the main identified hindrances.

The benefits we expect from our proposal are mainly in terms of applicability. We pursue with this specific approach to minimize the stakeholder’s implication without loss of quality on their contributions. That expects enhancing the commitment of higher management
### Incremental Case Study

<table>
<thead>
<tr>
<th>STEP</th>
<th>PURPOSE</th>
<th>SCOPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRST</td>
<td>Feasibility</td>
<td>Simplified application of the aggregation strategy oriented to proof the concept. Easily accessible measures and assessment. Simple preprocessing and computation.</td>
</tr>
<tr>
<td>SECOND</td>
<td>Practicability &amp; User Perception</td>
<td>Complete application of the aggregation strategy oriented to satisfy real industrial need. Complete pre-processing and data consistency assurance. Limited to a sole reference point</td>
</tr>
<tr>
<td>THIRD</td>
<td>Industrial applicability From attributes to behaviour</td>
<td>Extended application to two reference points at relevant product releases. Integration of the low level attributes and Usage Model concept. Integration of project management records and corrected defects between releases. Analysis from low level attributes to exhibited behaviour. Applicability assessment</td>
</tr>
</tbody>
</table>

**Table 5.6: The Incremental Case Study**

levels with the quality assurance programs since requiring reduced resources thus increased chances of integration in daily industrial practices.
CHAPTER 6

THE INCREMENTAL CASE STUDY.

The industrial application is not only the preferential Validation Strategy for any proposal in the Software Engineering field but also the ultimate objective of whatever Research and Development effort in any engineering discipline. This chapter is, thus, devoted to the application of the proposed SB-SRM framework in order to demonstrate the feasibility of the approach as well as its utility as analysis methodology. To achieve that, a large real-world Case Study has been defined and carried out along two years of monitoring and data gathering. Additionally the proposed Incremental Case Study strategy simultaneously underwent its own validation cycle. The two first steps on this research were already presented in [43] and [42].

_The facts are always less than what really happened._

Nadine Gordimer

6.1. THE CASE STUDY: DEFINITION AND DATA GATHERING.

This chapter shows the study conducted to investigate the efficiency and effectiveness of SB-SRM in an industrial setting. We will put a particular focus to; on one hand on the user-based quality and our ability to capture the stakeholders view on the topic and on the other hand on how the relationship between source code attributes and system exhibited behaviour can be analysed in a straightforward way. In practice this means that the overall goal in this Case Study is to assess in which extent the proposed model and application methods met its intended purpose giving an answer to those hindrances that today hampers a broader application of the SB-SRM within common industrial constraints.

This overall objective expected to be achieved by means of the Case Study is refined into a set of research questions, which are to be answered through the case study analysis. We
refine this high level objective to more specific goals by adapting the standard template of the well-known Goal-Question-Metric [10, 82] as follows:

**The overall goal** in this Case Study is to: **Analyze** the application of a structural model to an existing software system for the purpose of software Reliability evaluation with respect to its effect on Software Reliability, development and maintenance efficiency from the point of view of the system technical officers in the context of a large legacy real-time industrial system.

With the purpose to promote later analysis of the Case Study outcomes such overall goal needs to be expressed in a narrowed way. This research goal was then broken down into the following more specific questions, see table 6.1, stating what is needed to know in order to fulfill the goal of the study.

<table>
<thead>
<tr>
<th>RQ1</th>
<th>How can we assess Reliability?</th>
<th>Motivation: Contribute to produce better software product.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ2</td>
<td>Which are its components?</td>
<td>Motivation: Contribute to improve software reliability knowledge.</td>
</tr>
<tr>
<td>RQ3</td>
<td>How do the components relate to one another?</td>
<td>Motivation: Evaluate the descriptiveness of the proposed model on real world context</td>
</tr>
<tr>
<td>RQ4</td>
<td>How the proposed structural model is applicable in daily industrial practice?</td>
<td>Motivation: Identify the needs/issues/problems that can be addressed using the proposed SB-SRM analysis within the context of developing and running software-intensive systems.</td>
</tr>
</tbody>
</table>

**Table 6.1:** Research Questions and motivation

Concerning the application protocol and as previously explained in chapter 5 we will proceed to apply such methodology as an Incremental Case Study of three steps of successive larger scope. First one, as a proof-of-concept will seek for feasibility confirmation, second one will cope with the practicability assessment and User Perception of Product Reliability and the third one with the larger scope of its industrial applicability, stakeholders’ needs satisfaction and the link between the low level product’s attributes and the system exhibited behaviour.
Case Study Definition

Case Description. To develop the Case Study we have used a dataset collected from a lead European Space Industry Company (ESIC, real name concealed for confidentially reasons) and from of a large legacy real-time industrial software system. The software product under study is about fifteen million lines of code implemented as a middleware based open system and running in a distributed architecture of about one hundred nodes. Characterized by a high heterogeneity of sub-systems and technologies; Linux/Corba, web-based services, COTS, C++, Java, C, Fortran, TCL, Python, Shell, it makes part of a larger Real Time Monitoring & Control System (MCS).

The system undergoes continuous functional modifications (requirements are added and removed) in response to the dynamic business environment. In addition to user visible functionality other typical modifications include adaptations to changes in law and regulations (e.g. recently those concerning the custody of personal records) or in local working routines (e.g. automated reporting on data management activities) as well as corrective maintenance triggered by end user’s error reports. ESIC applies a classical waterfall process model for software maintenance. Each software release starts with a Statement of Work (SoW) defined by stakeholders representatives and product managers at the organization. Based on the SoW, software development process goes through requirements specification, design, implementation and unit testing, integration testing, user acceptance and non-regression testing, and then deployment to exploitation environment. The maintenance process is formally qualified by ISO 9001 and ISO 27000 and supported by European Standards ECSS 1.

The Software development process is outsourced under the overview and direction of ESIC engineers acting as product managers. The providers deliver the product once after successfully tested on factory. On site testing focus on delivered functionality and operational consistency. Since the Software modification is not driven by a need of increasing Reliability but mainly by functional adaptation to the highly dynamical business environment, the purpose of on-site integration testing is not testing till reaching a predefined level of Reliability, but to assuring that the achieved Reliability level is not degraded but increased in the extent of possible.

1European Cooperation for Space Standardization
In summary, the business model run by ESIC being very dynamic and adaptive to external constraints and market evolution, leads to a highly dynamic Software Maintenance Process built over the principle of continuous build-release cycles. Also, being this process fundamentally outsourced very detailed project records are hold, which will allow their exploitation as a reliable data source for our study in despite of not having being designed to such purpose.

**Case Definition.** The study will consist in the assessment of the system behavior as perceived by a representative group of stakeholders at two relevant moments, $T_0$ and $T$ with $T_0 < T$, analyzing how the modifications introduced into the system relates to the exhibited behavior. In the meantime, system operation reports will be monitored and system usage will be modelled. Also historical management records will be exploited in order to follow up the software configuration all along the study time span of roughly two years from $T_0$ to $T$.

$T_0$ will correspond to the operational deployment of a mayor selected release and $T$ will correspond to another one. The modifications included into the system between $T_0$ and $T$ were thoroughly analyzed in order to collect and classify those modifications with potential impact on the under study reliability characteristics. The final purpose is to characterize those system modifications in a way that permits to model its impact in the global Reliability perception then to compare these results with those obtained from the stakeholders' assessment.

This design is intended to permit the empirical validation of the proposed SB-SRM model by means of a real-world application but also to provide a path for researching on the relationships between software product low-level attributes and the higher level Reliability Sub-characteristics. The hierarchical model under validation is the one proposed and justified on Chapter 4 which constitutes the object of this research.

A panel of experts was formed based on their knowledge, skills and experience on the maintenance and operation of the aforementioned system. The participants involved are three Technical Managers from the ESIC - IT department and six senior professionals from software development division within an Engineering Company which is a major player in the global Space & Defense European market and partner of ESIC on MCS maintenance. 

---

2 acting as domain experts
Aged between 40 and 50 years, they all hold high college degrees in computer-related areas and have from 15 to 20 years of IT managing experience at different levels.

In practice, little before $T_0$ the expert panel was requested on its assessment about the performance of the reliability characteristic as per the model (Chapter 4) in order to get a first reference point. Some weeks after $T$, the exercise was repeated again in order to get a second reference. The expert panel was also requested on their assessment on the relative importance of the reliability characteristics in a time very different (few months) of $T_0$ with the intention to avoid bias by association of this one with the first inquiry. With the same purpose the group of experts was never informed of the final goal of the study or on how their provided data will be exploited as they were also instructed on the need of not discussing among them the provided assessment. Still to avoid bias the panel was not aware of the model layout but only of the Square definitions of Reliability and its selected sub-characteristics. All the individuals were, notwithstanding, informed and agreed on this on-purpose lack of information.

From this Case definition we derive the required data to be collected as well as the process to do it.

**Case Study Data Gathering**

The information needed to develop the model consists of two parts: those data required to parameterize the model and those to apply it to our particular case, being this last of two different natures, on one hand expert subjective assessments and on the other objective data elicited from system operation logs and project management record. The process of data collection consists then of two categories, as is shown in figure 6.1, those data which are required to develop and parameterize the structural model and on the other hand to gather those data which are inputs to the model application.

The majority of information is then gathered from historical records and domain experts.

1. **Domain Experts**: Data from experts panel will be exploited for both, building up the model and its application to a case. For the first purpose we requested them on the relative importance of model elements deriving from assessment the model parameters. For the application, we will need their inputs in the form of performance
2. **Project management Records**: Will be exploited as primary source of data for low level system attributes and usage modeling. In particular to identify software defects and its impact on system behavior related to Reliability.

3. **System Operation Logs**: Will be used as secondary information source for the System Usage Modelling. In particular to evaluate the frequency of use of each of the system areas when not clearly deduced from the primary source, the Project Records.

Project Management Records is a relevant example of third degree data which refers to, for example, minutes of meeting, documents from different development phases, organizational charts, financial records, and previously collected measurements in an organization. As for other third degree data sources it is important to keep in mind that the documents were not originally developed with the intention to provide data to research in a case study.

A particular strength of the data collected for this study concerns the fact that none of the logs or records was designed for the study but is pre-existent data, also none of the expert in the panel was informed on the purpose of the assessment they were providing thus no bias is to be expected regarding that. We consider credibility of the collected data very high for these reasons. We also consider the data collected very reliable for the following reasons:
1. The participant experts are extremely familiar with the MCS technology and behavior as well as with ESIC culture and business model.

2. The dataset for this study was collected non-invasively. This way, the system users were not distracted from their daily work and they did not change their usual behavior because of the ongoing study.

Regarding data validity and completeness, during our study only one of the participants decided to partially stop his participation at the second assessment. Nevertheless, the data from the rest of them is still complete enough as to build a second consensus. In that, the strengths of the AHP technique revealed of very valuable support. Data validity is also improved by the application of some of the recommended [104] practices for improving validity in case studies in software engineering:

1. Member checking: Go back to research participants to ensure that the interpretations of the data make sense from their perspective.

2. Rich, thick descriptions: Where possible, use detailed descriptions to convey the setting and findings of the research.

3. Prolonged contact with participants: Make sure that exposure to the subject population is long enough to ensure a reasonable understanding of the issues and phenomenon under study.

In our application points 1 and 2 are ensured by the documents provided to the experts as well as by the direct access to the main researcher in order to provide any required clarification. The participants were asked to provide their assessment on the basis of the aforementioned definitions after been instructed on the intentions and on the significance of the assessment scale, notwithstanding without awareness on the specific model layout, this in order to avoid bias or influencing them on their judgement and so improving the empirical validity. Point 3 is ensured by the selection of the panel members whom as explained have a professional experience largely above ten years on the developing and maintenance of the under-study Software Product and its exploitation context.
Gathering the Field Data. In practice each kind of data was obtained by means of specific dedicated process as we will see on the following.

1. Data from domain experts was gathered by direct interview between this thesis author and each of the experts taking part on the study.

2. Data from historical records was extracted by the main researcher who has access to it and the domain knowledge to interpret them.

From Domain Experts: The expert panel was asked to make pair-wise assessments of the relative importance between each pair of sub-characteristics in the SB-SRM model. This comparison is in the form of linguistic variables and was done after instruction from the principal researcher on how interpreting the assessment scale. That leads to the agreement on linguistic variables to use thus promoting the consistency between the assessments from each individual. Similar evaluation is also made but on system performance regarding each of the Reliability sub-characteristic and the overall Reliability itself. This second bunch of interviews was performed in a different time to avoid bias on the empirical data due to a undesired learning or feedback effect.

From Historical Records: The more relevant is the project records mining in order to obtain the necessary data on Software Defects which is the low level Software Product attribute which we are intended to relate to the high level Quality Characteristics and Sub-characteristics. This will be further detailed in section 6.2.1 and 6.2.3 cosidering its context. In addition, several product metrics where directly obtained from project records; the number of modules modified in the under assessment software release as well as the availability figures. This data was mainly intended for proof-of-concept purposes as we will see on the following section.

6.2. THE CASE STUDY: INCREMENTAL APPLICATION AND ANALYSIS

As detailed in section 5.2 the first step in any Analytic Hierarchy Process is that the problem has to be modeled into a hierarchy i.e. the one 6.2 we have proposed in Chapter 4, which is
fundamental to the process of the FAHP.

<table>
<thead>
<tr>
<th>RELIABILITY:</th>
<th>Maturity</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Degree to which a system, product or component meets stakeholder’s needs under normal operation.</strong></td>
<td><strong>Degree to which a system, product or component is operational and accessible when required for use.</strong></td>
<td></td>
</tr>
<tr>
<td>Stability</td>
<td>Robustness</td>
<td>Fault Tolerance</td>
</tr>
<tr>
<td><strong>Degree to which a system, product or component requires of modifications after release as result of defects presence or because of new, missing or wrong requirements.</strong></td>
<td><strong>Degree to which a system or component can function correctly in the presence of invalid inputs or a stressful environment.</strong></td>
<td><strong>Degree to which a system, product or component operates as intended despite the presence of hardware or software faults.</strong></td>
</tr>
</tbody>
</table>

Figure 6.2: Software Reliability Structural Model

Hierarchy indicates, figure 6.2, a relationship between elements of one level with those of the level immediately below. Linguistic assessment is also collected from experts as explained in previous section 6.1. To do that we do not need the Hierarchy but just its structural elements. Also objective software product attributes is elicited from project records or the source code itself. Those collected data correspond to the leaves on the hierarchy. Their intended objective is twofold, on the one hand data will be applied to parameterize the model on the other to carry out a series of specific Reliability assessment, each of them with a different set of input data.

The first purpose, parameterizing the model, is achieved comparing the hierarchy elements to one another with respect to the impact on their parent node in the hierarchy and on the other as explained on Chapter 5. Once this is done we will run three applications to Reliability analysis of different scope and objective; from a simple academic proof-of-concept to a complete analysis in industrial context.

**6.2.1. The first incremental step: Proof-of-Concept and Feasibility.**

In order to validate whether and how the proposal can be easily developed in practice we have performed the first step on our incremental study. The objective in this reduced scope is to get:
6.2. THE CASE STUDY: INCREMENTAL APPLICATION AND ANALYSIS

- A first evaluation on the requirements on resources to run the intended analysis.
- Potential disturbances to the system operation.
- The difficulties to get the required data.
- The ease to expressing the outcomes in a way understandable by non-experts audiences.

This exercise consists of a simple but quite standard application of the recognized and widely used Analytical Hierarchy Process (AHP) as described in Chapter 5. As inputs to the assessment we will use both, expert judgments and analytical objective measures.

**The AHP Method Application**

For this initial step the linguistic assessment is translated onto a crisp rational scale. This assessment consisted of the pairwise comparison regarding the relative relevance of each of the reliability’s sub-characteristics on our proposed layout. For the specific purpose of this step, answers are just combined by standard averaging. Table 6.2 sum-up the assessment of the experts, where X/Y means that the characteristic on the row is X/Y times more important than the one on the column i.e. Availability is 7/3 more important than Recoverability, and conversely Recoverability is 3/7 more important than Availability.

<table>
<thead>
<tr>
<th></th>
<th>Maturity</th>
<th>Availability</th>
<th>F. Tolerance</th>
<th>Recoverability</th>
<th>Robustness</th>
<th>Stability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maturity</td>
<td>1</td>
<td>13/9</td>
<td>5/2</td>
<td>5/2</td>
<td>17/12</td>
<td>13/3</td>
</tr>
<tr>
<td>Availability</td>
<td>1</td>
<td>7/3</td>
<td>7/3</td>
<td>11/6</td>
<td>7/5</td>
<td></td>
</tr>
<tr>
<td>F. Tolerance</td>
<td>1</td>
<td>19/12</td>
<td>7/3</td>
<td>9/5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recoverability</td>
<td>1</td>
<td>7/3</td>
<td>4/5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robustness</td>
<td></td>
<td></td>
<td>1</td>
<td>29/16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 6.2: Pairwise comparison*

These pairwise comparisons are arranged into the comparison matrix at each level and according to the hierarchy structure. Then, the vector representing the relative weights, or local priorities, of each of the attributes, can be calculated by computing the normalized eigenvector corresponding to the maximum eigenvalue of each matrix. This vector can be computed by dividing each element of the matrix with the sum of its column, then averaging the values in each row. Table 6.3 presents these local priorities on the left side, where
$w_x$ accounts for the priority or relative relevance of each characteristic regarding the one on their parent node e.g. $w_M$ accounts of the local priority for Maturity and $w_{Ft}$ for Fault Tolerance.

To aggregate the priorities along the hierarchy we have to consider that local priorities are calculated among same level elements and global ones are the product of all local weights from leaf to root on the hierarchy. In other words, the local priorities correspond to the relative weights of the nodes within a group of siblings with respect to their parent. Global priorities are obtained by multiplying the local priorities of the siblings by their parent’s global priority i.e. the global priority for Stability is $W_{St}$ computed as the product of $w_{St}$ and $w_M$ which are the local priorities on the path from the leaves to the root of the hierarchy. Table 6.3 sums-up this computation.

<table>
<thead>
<tr>
<th>Local Priorities</th>
<th>Global Priorities</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_M = \frac{13}{22}$, $w_{St} = \frac{15}{44}$, $w_{Rb} = \frac{29}{44}$</td>
<td>$W_{St} = 0.2015$, $W_{Rb} = 0.3984$</td>
</tr>
<tr>
<td>$w_A = \frac{9}{22}$, $w_{Ft} = \frac{19}{31}$, $w_{Rc} = \frac{12}{31}$</td>
<td>$W_{Ft} = 0.2507$, $W_{Rc} = 0.1584$</td>
</tr>
</tbody>
</table>

*Table 6.3: Computed Priorities*

In that simple, but recognized and well-founded way, we are capturing the view on Reliability for the particular group of stakeholders for which this expert’s panel is representative. It is also easy to present this view to non-expert stakeholders in a graphical way without information loss e.g. by means of a Kiviat’s diagram as we can see on Figure 6.3, where ST means Stability, RB Robustness, FT Fault Tolerance and RC Recoverability.

![Figure 6.3: Stakeholders View on Reliability](image)

This figure expresses in an easy to understand manner how, for the consulted group of experts, the more relevant sub-characteristic is the system Robustness, roughly three times...
more important than the system Recoverability which is consistent with the Organization Business model based on a continuously manned system and highly dynamic evolution thus Stability is less relevant as a gauge of Reliability than the Robustness which is quite important to enable such dynamic environment.

At that point, and to produce a reliability assessment showing how this method provides a simple way to incorporate experts’ subjective values together with factual data we have requested to the same panel their subjective evaluation $E_{FT}$ & $E_{RB}$ on the Fault Tolerance and Robustness Reliability characteristics as they consider are exhibited by the system. We have, also, defined two analytical metrics; $\sigma$ for the system stability based on the Maturity Index proposed on IEEE 982.1-1988 and $\lambda$ for the Recoverability re-using the normalized Mean Recovery Time (MRT) as proposed on ISO/IEC 9126. Table 6.4 sums-up the computation of the Reliability index.

<table>
<thead>
<tr>
<th>Inputs definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma \equiv (mt - \delta m)/mt$</td>
</tr>
<tr>
<td>$mt$: total number of modules</td>
</tr>
<tr>
<td>$\delta m$: number of modified modules</td>
</tr>
<tr>
<td>$\lambda = 1 - TM/T$</td>
</tr>
<tr>
<td>TM: MRT</td>
</tr>
<tr>
<td>T: Total Time</td>
</tr>
<tr>
<td>Expert’s Estimations</td>
</tr>
<tr>
<td>$E_{FT} = 13/9$</td>
</tr>
<tr>
<td>$E_{RB} = 61/18$</td>
</tr>
<tr>
<td>Index Computation</td>
</tr>
<tr>
<td>$R = W_s \sigma + W_r E_{RB} + W_f E_{FT} + W_r \lambda$</td>
</tr>
<tr>
<td>$R = 0.1612 + 0.27 + 0.0724 + 0.1108 = 0.6144$</td>
</tr>
</tbody>
</table>

Table 6.4: Reliability Computation

Finally, such outcome could also be presented as a Kiviat’s diagram, as in figure 6.4, which allows a joint description of the ideal objective and the actual assessment which is greatly helpful for the decision makers.

In the presented case it is clear that the estimated Robustness an even more the Fault Tolerance are far from this group of stakeholders’ expectations while the rest of characteristics are rather close to user’s requirements.

That is a pointer to consider in-depth analysis of system behavior in terms of Fault Tolerance,

\(^3\text{Which need to be normalized to allow comparisons}\)
assessing whether the system need a particular improvement on this aspect or it is just that such stakeholders expectations are not in line with the Organization’s Business objectives and constraints. In both cases the data have been translated into valuable information for the decision makers, which is the ultimate goal.

**First Incremental Step Results Analysis**

With this first step we got a partial answer to one of our research questions; How the proposed structural model is applicable in daily industrial practice? in the extent that this exercise has proven the feasibility of the proposed approach since validating the limited impact on the operational processes but allowing capturing the different views in a fairly simple manner. Addressing, in that way although in a limited extent, some of the main difficulties involved in Software Reliability Modeling application in industrial environments, the conceptual complexity and the stakeholders’ standpoint.

Coming back to the objectives stated at the beginning of this step, the conclusion is that the requirements on resources have been minimal and no disturbances to the system operation have been noted. Regarding the difficulties to get the required data, those we have choose for this first approach are not particularly complex and of ease gathering, but more evidence is still required from the next steps on this point.

Another objective was about the ease of expressing the outcomes in a way understandable by non-experts audiences. This kind of graphical representation of the outcomes what make
it accessible to non-experts users is very convenient in industrial context where relevant information is to be shared with very different stakeholders.

Notwithstanding this simple application of AHP has well-known drawbacks when dealing with qualitative human-made assessment, as discussed in Chapter 5. Also the simple metrics we have applied, mainly the maturity index lack of enough descriptiveness as to inform the management of the system improvement, or could be even completely meaningless as discussed in [69] or can be easily mistaken as is explained in [121].

Another important issue that is not always taken into account is that Reliability is also a perception (maybe mainly a perception) on how the system behaves, as it is not being properly considered in this first step oriented to prove feasibility. We will cover this on the next step enlarging the scope of the exercise by considering the user perception dimension as well as applying a more sophisticated aggregation strategy better suited to deal with vagueness of human judgments.

6.2.2. The second incremental step: Users Perception of Reliability.

Once feasibility has been proved in the previous step the objectives in this enlarged scope are to get:

- To validate the industrial applicability of the proposed structural SB-SRM model

- To validate the conceptual descriptiveness of the proposed structural SB-SRM model

We will do that by analysing on which extent a group of users consider that the product meets their expectations in terms of exhibited Reliability. This scenario corresponds to another of the main hindrances identified as outcome of the Literature Review; the inability to capture the stakeholders’ expectations. Investigating means of capturing this user-based dimension of Software Product Reliability is important in order to increase end-user satisfaction with the delivered products which is a recognized competitive asset.

Computation Method Application

The six steps method, as presented in Chapter 5, is to be developed as follows:
Step 1: Linguistic Variables agreement
This step makes part of the initial phase of our Incremental Case Study Method. Linguistic Variables are necessary to the experts to provide their assessment. These are provided in the form of two Linguistic scales, one for comparisons in terms of relative importance and another one for assessment in terms or performance. It is important that all participants agree on the meaning of such scales. This agreement is ensured by personal interviews between this Thesis author and the external participants.

Step 2: Translation from Linguistic variables to TFNs
From the gathered field data we got a set of assessments from the domain experts like the ones from participant "A" that we will use on the following to shown how the F-AHP is applied.

As explained in Chapter 5 an appropriate mapping between linguistic variables and fuzzy numbers is required to apply the FAHP computations. Table 6.5 shows two linguistic scales we have used in this Case Study making explicit the flexibility of this methodology which constitutes one of its more appreciated assets.

<table>
<thead>
<tr>
<th>Linguistic variable</th>
<th>Triangular Fuzzy</th>
<th>Linguistic variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equally important</td>
<td>1,1,1</td>
<td>Least Relevance</td>
</tr>
<tr>
<td>Intermediate values</td>
<td>1,2,3</td>
<td>Intermediate values</td>
</tr>
<tr>
<td>Moderately important</td>
<td>2,3,4</td>
<td>Low Relevance</td>
</tr>
<tr>
<td>Intermediate values</td>
<td>3,4,5</td>
<td>Intermediate values</td>
</tr>
<tr>
<td>Essentially important</td>
<td>4,5,6</td>
<td>Normal Relevance</td>
</tr>
<tr>
<td>Intermediate values</td>
<td>5,6,7</td>
<td>Intermediate values</td>
</tr>
<tr>
<td>Very strong importance</td>
<td>6,7,8</td>
<td>High Relevance</td>
</tr>
<tr>
<td>Intermediate values</td>
<td>7,8,9</td>
<td>Intermediate values</td>
</tr>
<tr>
<td>Absolutely important</td>
<td>8,9,10</td>
<td>Extreme Relevance</td>
</tr>
</tbody>
</table>

Table 6.5: Fuzzy Assessment

The first operation to produce on this data is, then to convert them to Triangular Fuzzy Numbers.

Step 3: Comparison Matrix
Then, each participant assessment is arranged into a set of comparison matrix, as an example the following ones corresponds to the processing for participant "A", presented in Table 6.6 and Table 6.7 in its final form after re-arrangement according to the structural model under
study. The same computation is done for all the panel of participants.

<table>
<thead>
<tr>
<th>A</th>
<th>Maturity</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maturity</td>
<td>1,1,1</td>
<td>5, 6, 7</td>
</tr>
<tr>
<td>Availability</td>
<td>1/7,1/6,1/5</td>
<td>1,1,1</td>
</tr>
</tbody>
</table>

**Table 6.6: Assessment example L1**

That is, for expert "A" Availability is more relevant than Maturity but not in an extent as highly more relevant. This vague (but accurate) assessment is coded as intermediate among them, thus (5, 6, 7). The ability of doing this constitutes one of the major assets of the FAHP methodology. In an analogue way the assessment for the rest of subcharacteristics is expressed on the following.

<table>
<thead>
<tr>
<th>A</th>
<th>F.Tolerance</th>
<th>Recoverability</th>
<th>Robustness</th>
<th>Stability</th>
</tr>
</thead>
<tbody>
<tr>
<td>F. Tolerance</td>
<td>1,1,1</td>
<td>3,4,5</td>
<td>1/3,1/2,1</td>
<td>1,1,1</td>
</tr>
<tr>
<td>Recoverability</td>
<td>1/5,1/4,1/3</td>
<td>1,1,1</td>
<td>1/4,1/3,1/2</td>
<td>3,4,5</td>
</tr>
<tr>
<td>Robustness</td>
<td>1,2,3</td>
<td>2,3,4</td>
<td>1,1,1</td>
<td>2,3,4</td>
</tr>
<tr>
<td>Stability</td>
<td>1,1,1</td>
<td>1/5,1/4,1/3</td>
<td>1/4,1/3,1/2</td>
<td>1,1,1</td>
</tr>
</tbody>
</table>

**Table 6.7: Assessment example L2**

**Step 4: Consistency checking.**

Those data need to be reviewed in order to ensure the internal consistency of each of the provided set of pairwise comparison. Inconsistent data will be rejected. T.L. Saaty [108, 109] proved that for consistent reciprocal matrix, the largest eigenvalue $\lambda_{max}$ is equal to the size of the matrix, $\lambda_{max} = n$ for $n > 2$. Then he gives a measure of consistency, called Consistency Index as deviation or degree of consistency using the following formula:

$$CI = \frac{\lambda_{max} - n}{n - 1}$$

This largest or principal eigenvalue is obtained from the addition of products between each element of the eigenvector and the sum of columns of the reciprocal matrix. In this approach, each triangular number is transformed to a non-fuzzy number, we choose the modal value just to simplify the computation. The eigenvector is obtained by dividing each element by the sum of its column then adding by rows and normalizing by the matrix size. Finally, the consistency ratio (CR) is defined as a ratio between the consistency of a given evaluation matrix and consistency of a random matrix:
\[ CR = \frac{CI}{RI(n)} \] (6.2)

Where RI(n) as it was explained in subsection 5.2.2. is a random index that depends on matrix dimension "n", as shown in Table 6.8.

<table>
<thead>
<tr>
<th>n</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>RI</td>
<td>0.58</td>
<td>0.9</td>
<td>1.12</td>
<td>1.24</td>
<td>1.32</td>
<td>1.41</td>
<td>1.45</td>
</tr>
</tbody>
</table>

Table 6.8: Random Index

Applying the above to the first of our comparison matrix, Table 6.7, we obtain the following normalized comparison matrix:

\[
\begin{pmatrix}
0.2 & 0.14 & 0.36 & 0.285 \\
0.6 & 0.428 & 0.36 & 0.285 \\
0.1 & 0.214 & 0.181 & 0.285 \\
0.1 & 0.214 & 0.09 & 0.142
\end{pmatrix}
\] (6.3)

From which we compute the eigenvector: \( W = (0.24265, 0.41825, 0.0195, 0.01365) \)

Then the principal eigenvalue is: \( \lambda_{max} = 4.2018 \)

From where \( CI = 0.06726 \) as per equation (1) and finally \( CR = 0.0747 \leq 0.1 \) as per equation (2) Thus the assessment from contributor A is consistent enough as to be considered in the analysis. The same was done for the rest of the contributions obtaining similar values. All contributions were accepted as input to the model, thus entered in the next step, which is the aggregation of the individual assessments into a unique set named "consensus".

**Step 5: Synthesis, Aggregation and consensus.**

The objective at this stage is producing a single set of comparison matrix which takes into consideration each participant’s assessment. There is a variety of proposal available in the literature for the aggregation of every participant assessment into a single consensual judgement matrix, we chose for the sake of simplicity the geometric mean as explained

\footnote{This was not surprising since all contributors share similar profile and long experience collaborating together in MCS maintainence and evolution}
in [29, 30] which is one of broad application. In summary this approach consists in, for triangular fuzzy numbers in the form \((l_i, m_i, u_i)\), choosing the minimal value among the lower elements \(l_i\), the maximal among the \(u_i\) upper elements and the geometric mean of the \(m_i\) modal values.

**Step 5.1 Aggregated Comparison Matrix**

We proceed in that way and, taking into consideration the layout of the proposed SB-SRM, we obtain the following set of matrix, tables 6.9, 6.10 and 6.11, which sum-up the stakeholders assessment for the structural elements of the under analysis model.

<table>
<thead>
<tr>
<th>L1</th>
<th>Maturity</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maturity</td>
<td>1,1,1</td>
<td>0.25, 1.59, 5</td>
</tr>
<tr>
<td>Availability</td>
<td>0.20, 0.69, 4</td>
<td>1,1,1</td>
</tr>
</tbody>
</table>

*Table 6.9: Firs Level Assessment*

<table>
<thead>
<tr>
<th>L2a</th>
<th>Robustness</th>
<th>Stability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robustness</td>
<td>1,1,1</td>
<td>2, 3, 4</td>
</tr>
<tr>
<td>Stability</td>
<td>0.25, 0.33, 0.5</td>
<td>1,1,1</td>
</tr>
</tbody>
</table>

*Table 6.10: Second Level Assessment. Left branch*

<table>
<thead>
<tr>
<th>L2b</th>
<th>F.Tolerance</th>
<th>Recoverability</th>
</tr>
</thead>
<tbody>
<tr>
<td>F. Tolerance</td>
<td>1,1,1</td>
<td>0.33, 1.26, 5</td>
</tr>
<tr>
<td>Recoverability</td>
<td>0.2, 0.79, 3</td>
<td>1,1,1</td>
</tr>
</tbody>
</table>

*Table 6.11: Second Level Assessment. Right branch*

Where L1 denotes the conceptual Level 1 and L2a and L2b both branches on the second level of the hierarchy depicted on table 6.2.

**Step 5.2 Model parameterization**

Now we can compute the model parameters, commonly referred as the fuzzy weight vectors. Yet for the sake of simplicity, by using geometric mean technique [30, 97]. First, geometric mean is computed by rows and then it is normalised to allow proper comparisons.

In our model it is particularly simple since the proposed layout leads to small matrix as we have seen before. Then those triangular numbers are translated onto crisp or “de-fuzzified” values, we will apply the simple centroid method. Then, those crisp values are normalized. Results are in table 6.12.
Finally we obtain the complete structure of relative importance of each element in our structural model accordingly to stakeholder point of view as it is show in table 6.13. Those are the model parameters for this case of application.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Fuzzy vector</th>
<th>Crisp value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maturity</td>
<td>(0.118, 0.603, 2.35)</td>
<td>0.694</td>
</tr>
<tr>
<td>Availability</td>
<td>(0.105, 0.397, 0.851)</td>
<td>0.3059</td>
</tr>
<tr>
<td>Robustness</td>
<td>(0.52, 0.752, 1.05)</td>
<td>0.7429</td>
</tr>
<tr>
<td>Stability</td>
<td>(0.184, 0.248, 0.372)</td>
<td>0.257</td>
</tr>
<tr>
<td>F. Tolerance</td>
<td>(0.229, 0.58, 2.193)</td>
<td>0.6201</td>
</tr>
<tr>
<td>Recoverability</td>
<td>(0.179, 0.42, 1.239)</td>
<td>0.3798</td>
</tr>
</tbody>
</table>

**Table 6.12: The Global Weight Vector**

**Step 6: Model Application**

Once the structural model is parameterized with the global weights in 6.13 we can use it to evaluate the user perception of the Software Reliability for the system under study. To do so, we need the expert assessment which was also collected as a linguistic variable. In table 6.14 the provided answers after standard linguistic variables translated into Triangular Fuzzy Numbers are shown. Those data are summed-up into a unique global assessment, consensus, in the same way as it was previously done for the relevance parameters. The outcome is in the following table 6.15.

<table>
<thead>
<tr>
<th>Local Weights</th>
<th>Global Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>(w_M = 0.694) (w_{St} = 0.257) (w_{Rb} = 0.7429)</td>
<td>(W_{St} = 0.1783) (W_{Rb} = 0.5155)</td>
</tr>
<tr>
<td>(w_A = 0.3059) (w_{Ft} = 0.6201) (w_{Rc} = 0.371)</td>
<td>(W_{Ft} = 0.1897) (W_{Rc} = 0.1135)</td>
</tr>
</tbody>
</table>

**Table 6.13: Computed Relative Importances**

<table>
<thead>
<tr>
<th>Reliability</th>
<th>F. Tolerance</th>
<th>Recoverability</th>
<th>Robustness</th>
<th>Stability</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6,7,8</td>
<td>5,6,7</td>
<td>6,7,8</td>
<td>7,8,9</td>
</tr>
<tr>
<td>B</td>
<td>4,5,6</td>
<td>6,7,8</td>
<td>6,7,8</td>
<td>4,5,6</td>
</tr>
<tr>
<td>C</td>
<td>6,7,8</td>
<td>4,5,6</td>
<td>6,7,8</td>
<td>4,5,6</td>
</tr>
<tr>
<td>D</td>
<td>7,8,9</td>
<td>6,7,8</td>
<td>6,7,8</td>
<td>7,8,9</td>
</tr>
<tr>
<td>E</td>
<td>8,9,10</td>
<td>5,6,7</td>
<td>6,7,8</td>
<td>6,7,8</td>
</tr>
<tr>
<td>F</td>
<td>7,8,9</td>
<td>6,7,8</td>
<td>6,7,8</td>
<td>7,8,9</td>
</tr>
<tr>
<td>G</td>
<td>7,8,9</td>
<td>6,7,8</td>
<td>6,7,8</td>
<td>7,8,9</td>
</tr>
<tr>
<td>H</td>
<td>7,8,9</td>
<td>6,7,8</td>
<td>6,7,8</td>
<td>7,8,9</td>
</tr>
</tbody>
</table>

**Table 6.14: Users Quality Perception**

Values which are subsequently de-fuzzied by means of the centroid method to obtain an assessment expressed in crisp values, table 6.16, which we use as input to our proposed
6.2. THE CASE STUDY: INCREMENTAL APPLICATION AND ANALYSIS

<table>
<thead>
<tr>
<th>Reliability</th>
<th>F. Tolerance</th>
<th>Recover</th>
<th>Robustness</th>
<th>Stability</th>
</tr>
</thead>
<tbody>
<tr>
<td>4, 7.1, 10</td>
<td>4, 6.48, 8</td>
<td>6, 7, 8</td>
<td>4, 6.543, 9</td>
<td>5, 7.34, 9</td>
</tr>
</tbody>
</table>

**Table 6.15:** FAHP Consensus on User Perception

structural model.

<table>
<thead>
<tr>
<th>Reliability</th>
<th>F.Tolerance</th>
<th>Recoverability</th>
<th>Robustness</th>
<th>Stability</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.03</td>
<td>6.16</td>
<td>7</td>
<td>6.51</td>
<td>7.11</td>
</tr>
</tbody>
</table>

**Table 6.16:** De-Fuzzyfied Consensus

**Second Increment Results Analysis**

The stated objectives for this second step are to validate the industrial applicability as well as to validate the conceptual descriptiveness of the proposed structural SB-SRM model. We can, reasonably conclude that both objectives are attained.

Regarding the industrial applicability this second step has reaffirmed that the requirements on resources have been minimal and no disturbances to the system operation have been introduced. In the same vein no particular difficulties to get the necessary data have been noted. The fact that the computations are more complex is not to be considered as a impairment since they are easy to automate. Also they exists today FAHP tools from a variety of vendors.

From the obtained results, and in order to evaluate in which extent the proposed structural model is able to capture the stakeholder vision we compute Reliability from table 6.13 parameters and table 6.16 user assessments, then compare the result to the user consensus on Reliability recorded in table 6.16.

\[
R = W_{Ft} \times 6.16 + W_{Rc} \times 7 + W_{Rb} \times 6.51 + W_{St} \times 7.11
\]

\[
R = 0.1897 \times 6.16 + 0.1135 \times 7 + 0.5155 \times 6.51 + 0.1783 \times 7.11 = 6.6
\]

This outcome means that the deviation between the user perceived Reliability (7.03) on table 6.16 and the one computed through the model and proposed framework (6.6) is of roughly a 6%. In other words, in that simple, but recognized and well-founded way, we are actually capturing the view on Reliability for the particular group of stakeholders for which this experts panel is representative. We deem that it is also reasonable claiming that our second
objective is attained in this second step.

Notwithstanding a drawback of this exercise is that it does not address the issues concerning the difficulties to make the link between the product attributes and the system exhibited behavior. Next step thus includes the extension of the presented approach to cope with another relevant issue when analysing Software Product Quality; making the link between the very high nature of the Quality models and the low level Software Product properties and attributes in a sound, understandable and simple manner.

6.2.3. The third incremental step: From Attributes to Behavior

The main objectives on this, yet of enlarged scope, step can be developed as:

- Gaining empirical evidence on the suitability of the proposed model and method.
- Shown that the analysis can be applied on a continuous improvement process.
- Analysing the relationship between source code attributes and the exhibited behaviour.

For this final step we are thus going to consider three different areas to work on:

1. The user assessment at two different relevant moments in the project.
2. The System Usage Modeling and its exhibited behavior.
3. The Product Assessment synthetizing operational usage and manifested defects.

This step is built over the two precedents which have shown feasibility and parameterized the model as well as have proven that it is able to account of user view on product reliability.

Software Product Reliability evolution

The user’s assessment correspond to the analysis done in the previous step, which correspond to our $T_0$ initial reference that we will apply now to the moment T corresponding to our study endpoint. As a remainder, $T_0$ corresponds to a major release and the starting point for the study and T to another major release.
6.2. THE CASE STUDY: INCREMENTAL APPLICATION AND ANALYSIS

<table>
<thead>
<tr>
<th></th>
<th>Reliability</th>
<th>Maturity</th>
<th>Availability</th>
<th>F. Tolerance</th>
<th>Recoverability</th>
<th>Robustness</th>
<th>Stability</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6.7,8</td>
<td>5.6,7</td>
<td>6.7,8</td>
<td>7.8,9</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>5.6,7</td>
</tr>
<tr>
<td>B</td>
<td>4.5,6</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>4.5,6</td>
<td>6.7,8</td>
<td>4.5,6</td>
<td>6.7,8</td>
</tr>
<tr>
<td>C</td>
<td>6.7,8</td>
<td>4.5,6</td>
<td>6.7,8</td>
<td>4.5,6</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>4.5,6</td>
</tr>
<tr>
<td>D</td>
<td>7.8,9</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>7.8,9</td>
<td>7.8,9</td>
<td>6.7,8</td>
</tr>
<tr>
<td>E</td>
<td>8.9,10</td>
<td>5.6,7</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>5.6,7</td>
<td>8.9,10</td>
<td>5.6,7</td>
</tr>
<tr>
<td>F</td>
<td>7.8,9</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>7.8,9</td>
<td>7.8,9</td>
<td>6.7,8</td>
</tr>
<tr>
<td>G</td>
<td>7.8,9</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>7.8,9</td>
<td>7.8,9</td>
<td>6.7,8</td>
</tr>
<tr>
<td>H</td>
<td>7.8,9</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>7.8,9</td>
<td>7.8,9</td>
<td>6.7,8</td>
</tr>
<tr>
<td>conse</td>
<td>4, 7.1, 10</td>
<td>4, 6.48,8</td>
<td>6, 7, 8</td>
<td>4, 6.5439</td>
<td>5, 7.34, 9</td>
<td>4, 7.1, 10</td>
<td>4, 6.48,8</td>
</tr>
</tbody>
</table>

Table 6.17: Users Subjective Assessment at $T_0$

The same computations are then performed to get a consensual assessment on Reliability and its sub-characteristics performance. As before, the linguistic assessment is translated into Triangular Fuzzy Numbers for computation.

As previously mentioned, one of the committee members moved to other responsibilities and did not completed this task, abandoning the study before $T$. This is not an issue since once of the more recognized features of AHP methodology is its robustness to this kind of events, which are fundamentally the same that rejecting a set of assessment because of inconsistency vs Saaty Random Index.

<table>
<thead>
<tr>
<th></th>
<th>Reliability</th>
<th>Maturity</th>
<th>Availability</th>
<th>F. Tolerance</th>
<th>Recoverability</th>
<th>Robustness</th>
<th>Stability</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>7.8,9</td>
<td>5.6,7</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>6.7,8</td>
</tr>
<tr>
<td>B</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>5.6,7</td>
<td>5.6,7</td>
<td>6.7,8</td>
<td>7.8,9</td>
</tr>
<tr>
<td>D</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>5.6,7</td>
<td>6.7,8</td>
<td>5.6,7</td>
</tr>
<tr>
<td>E</td>
<td>7.8,9</td>
<td>6.7,8</td>
<td>7.8,9</td>
<td>7.8,9</td>
<td>5.6,7</td>
<td>5.6,7</td>
<td>5.6,7</td>
</tr>
<tr>
<td>F</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>5.6,7</td>
<td>6.7,8</td>
<td>5.6,7</td>
<td>5.6,7</td>
</tr>
<tr>
<td>G</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>7.8,9</td>
<td>5.6,7</td>
<td>6.7,8</td>
<td>7.8,9</td>
<td>6.7,8</td>
</tr>
<tr>
<td>H</td>
<td>6.7,8</td>
<td>6.7,8</td>
<td>7.8,9</td>
<td>5.6,7</td>
<td>5.6,7</td>
<td>7.8,9</td>
<td>6.7,8</td>
</tr>
<tr>
<td>conse</td>
<td>6, 7.13, 9</td>
<td>6,7,8</td>
<td>6,7,5, 9</td>
<td>5, 6.39, 9</td>
<td>5, 6.40, 8</td>
<td>5, 6.53, 9</td>
<td>5, 6.98, 9</td>
</tr>
</tbody>
</table>

Table 6.18: Users Subjective Assessment at $T$

We can now to evaluate from these assessments the perceived evolution of system quality as per the expert panel expressed opinion. We do so by simple subtraction of fuzzy consensual evaluation for initial $T_0$ reference from final consensus at time $T$.

From the comparison of these two assessments we can then conclude that from the stakeholders’ point of view the Stability and Fault Tolerance have had a positive evolution whilst Robustness and more clearly Recoverability have had a negative evolution. In qualitative
CHAPTER 6. THE INCREMENTAL CASE STUDY.

<table>
<thead>
<tr>
<th></th>
<th>$T - T_0$</th>
<th>$\Delta$</th>
<th>CRISP</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>(5, 6.98, 9)-(4, 6.48, 8)</td>
<td>(1, 0.5, 1)</td>
<td>0.83</td>
</tr>
<tr>
<td>Rb</td>
<td>(5, 6.53, 9)-(4, 7.1, 10)</td>
<td>(1, -0.56, -1)</td>
<td>-0.19</td>
</tr>
<tr>
<td>Ft</td>
<td>(5, 6.39, 9)-(4, 6.54, 9)</td>
<td>(1, -0.15, 0)</td>
<td>0.28</td>
</tr>
<tr>
<td>Rc</td>
<td>(5, 6.40, 8)-(5, 7.34, 9)</td>
<td>(0, -0.94, -1)</td>
<td>-0.65</td>
</tr>
</tbody>
</table>

Table 6.19: FAHP Consensus on Reliability evolution

terms from the better to the lower performance according to the expert’s assessment the reliability sub-characteristics as per our proposed layout evolved as; $S > Ft > Rb > Rc$ that is, positive for the sub-characteristics $S$ and $Ft$ stable or slightly negative for $Rb$ and rather negative for $Rc$.

**Software Product Behaviour Assessment**

The challenge now it is to relate the Software Product attributes *i.e.* lines of code, function points or defects in the source code to this subjective evaluation. This is of paramount interest since in the extent we will able to do it we could decide how to modify the Product in order to obtain a predictable improvement. To do so we need to take into consideration the way the system is actually used in exploitation activities and how the defects manifest itself as observable misbehaviour or not. We have, then, developed an Usage Model understood as an analytic description of the system exploitation patterns.

**Usage Modeling.** As explained in Chapter 5 the system usage is modelled by means of what we have called Usage Model. We have, then, developed an Usage Model understood as an analytic description of the system exploitation patterns, considering when, how and why each of the system facilities is required. We do that on the basis of considering the Business Model that is; the mission of the under study system, its functional areas which constitutes the technological expression of such mission as well as the critically and the timing-related characteristics of its utilization.

Producing such a model is a task informed by the business domain knowledge and the system operation as well as the expertise on the Software Product Implementation. This modelling, expressed on following table 6.5 has been carried out by this Thesis author whom reunites such expertise and knowledge

On figure 6.5 column labeled ”Buss” denotes the main functional areas as per high level
business needs. Under "Func" we make a first level of detail considering functional areas with the same purpose. Real identifiers for Business needs and functional areas are irrelevant for the analysis and have been concealed. Label "Tc" is about temporal constraints and "Oc" about operational constraints. These two columns capture the user needs when using the system as linguistic variables, which once translated into triangular fuzzy numbers $(l_i, m_i, u_i)$ are combined by means of geometric mean which median value $m_i$ is show in the column "geo". Is this value which has been used to decide the final structural relevance with the criteria of choosing the closest grade or the lower one in case of geometric mean being the middle value. This is a simplification taken in an attempt of avoid obfuscating the description with more complicated computations which are out of the scope of our present objectives.

The usage frequency dimension will be further captured, in a case by case manner, as an incremental modifier derived from the detailed analysis of the failure data and accordingly to the following table 6.20 which meaning is raising or downgrading the relevance assessment of one or two steps depending on the usage frequency.

The presented level of detail on functional splitting together with the frequency modifier is judged sufficient for the purposes of the present study. Notwithstanding the hierarchical split could be easily extended to more detailed functional areas and levels if required. It is also easy to fix and re-apply in case of expert’s misjudgments. This flexibility is considered

<table>
<thead>
<tr>
<th>Buss</th>
<th>Func</th>
<th>Tc</th>
<th>Oc</th>
<th>geo</th>
<th>Struct. Relevance</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>a</td>
<td>Extre</td>
<td>Extre</td>
<td>9</td>
<td>Extre Aa</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>Low</td>
<td>Normal</td>
<td>3.87</td>
<td>Low Ab</td>
</tr>
<tr>
<td>B</td>
<td>a</td>
<td>High</td>
<td>High</td>
<td>7</td>
<td>High Ba</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>High</td>
<td>High</td>
<td>7</td>
<td>High Bb</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>Normal</td>
<td>Normal</td>
<td>5</td>
<td>Normal Bc</td>
</tr>
<tr>
<td>C</td>
<td>a</td>
<td>Normal</td>
<td>High</td>
<td>5.92</td>
<td>Normal Ca</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>Low</td>
<td>Normal</td>
<td>3.87</td>
<td>Low Cb</td>
</tr>
<tr>
<td>D</td>
<td>a</td>
<td>Least</td>
<td>Normal</td>
<td>2.23</td>
<td>Least Da</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>High</td>
<td>Extre</td>
<td>7.93</td>
<td>High Db</td>
</tr>
<tr>
<td>E</td>
<td>a</td>
<td>Normal</td>
<td>High</td>
<td>5.92</td>
<td>Normal Ea</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>Low</td>
<td>High</td>
<td>4.58</td>
<td>Normal Eb</td>
</tr>
<tr>
<td>F</td>
<td>a</td>
<td>High</td>
<td>Low</td>
<td>4.58</td>
<td>Normal Fa</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>Normal</td>
<td>Low</td>
<td>3.87</td>
<td>Low Fb</td>
</tr>
<tr>
<td>G</td>
<td>a</td>
<td>Low</td>
<td>Low</td>
<td>3</td>
<td>Low Ga</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>Least</td>
<td>Low</td>
<td>1.73</td>
<td>Least Gb</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>Least</td>
<td>Least</td>
<td>1</td>
<td>Least Gc</td>
</tr>
</tbody>
</table>

Figure 6.5: Functional Usage Model
Temporal requirement | modifier
---|---
Several per minute | ++
Several per hour | +
Some in a day | -
Rarely in a day | - -
Rarely in a week | - -

**Table 6.20:** Frequency Modifiers

an asset of the presented approach.

**Defects and Behavior: Product assessment.** In order to found relationships between the defects in the source code and the exhibited behavior we have analyzed every change reason from $T_0$ to $T$ which correspond to a two year time-span, screening for changes reporting on actual defects potentially related to the system Reliability in the terms we have defined it for our model.

This data need to be extracted from a variety product management reports and documents which are not designed nor intended to support this particular accounting activity. No particular classification for the change reason related to software product attributes is neither being used by ESIC, we were thus obliged to prepare one *ad hoc* taxonomy able to support our analysis. On the contrary detailed description on the observed behavior, impact on system operations and technical root causes, implemented modifications and relevance in terms of criticality in a five grades rational scale are reported and archived. Although not oriented to the analysis we intended in this Case Study the Project Management Records and the available systems logs are very complete and detailed as to permit, after mining them, the extraction of the required relevant data.

On the following table 6.21 we detail which data and on which purpose are elicited from the project archived records.

In practice that means that for a case which corresponds to, let say, Bc which correspond to a structural relevance described as "normal" (table 6.5) and in a function which is required just some times per day and is a failure with limited exposure to the user, thus of low visibility the global assessment is obtained as the geometric mean of "Normal" and "Low". In a case Gc, of structural relevance "Least" but on a function which is requires several times per hour,
Table 6.21: Collected Operational Data

<table>
<thead>
<tr>
<th>Data</th>
<th>Reference</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage Model</td>
<td>As per table</td>
<td>Required to obtain the relevance of each event which will be further combined with its visibility to obtain the global impact on user perception</td>
</tr>
<tr>
<td>Modifier</td>
<td>As per table</td>
<td>Required to capture the &quot;frequency of use&quot; concept on each specific event</td>
</tr>
<tr>
<td>Defect</td>
<td>As per Taxonomy on Table</td>
<td>Enabling a systematic analysis on the relationship between low level attributes and the higher level Reliability sub-characteristics</td>
</tr>
<tr>
<td>Failure</td>
<td>As defined in section</td>
<td>To inform both, the exposed visibility and the Reliability sub-characteristics to which the event is impacting. When the fault is also clearly identified it constitutes a valuable data</td>
</tr>
<tr>
<td>Visibility</td>
<td>Business Domain</td>
<td>Depending on the structural relevance and each particular misbehaviour we will provide a linguistic assessment on the visibility for the end user</td>
</tr>
<tr>
<td>Fix</td>
<td>From project records</td>
<td>To inform on the impacted subcharacteristic</td>
</tr>
<tr>
<td>Subcharacteristic</td>
<td>Our proposed Model</td>
<td>To determine on which subcharacteristic each particular defect impacts we will be informed by the failure - misbehaviour and the way it was fixed</td>
</tr>
</tbody>
</table>

we would apply the modifier "+" to raise a level to "low" and if the failure is of high exposure to the user we should compute the global effect on the user perception as the geometric mean of "low" and "High"

Table 6.22: Defect Global Assessment

<table>
<thead>
<tr>
<th></th>
<th>Relevance</th>
<th>Misbehaviour</th>
<th>Visibility</th>
<th>geoTFN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bc</td>
<td>Normal</td>
<td>Proc Hung</td>
<td>Low</td>
<td>(2, 3.87, 6)</td>
</tr>
<tr>
<td>Gc+</td>
<td>Low</td>
<td>Wong Info</td>
<td>Normal</td>
<td>(2, 3.87, 6)</td>
</tr>
<tr>
<td>Aa-</td>
<td>High</td>
<td>Proc Crash</td>
<td>High</td>
<td>(6, 7.0, 8)</td>
</tr>
</tbody>
</table>

In a similar way for a defect located in an area Aa but which is required only few times in a week we will downgrade the "Extreme" assessment corresponding to Aa of a step to "High" as shown in the table 6.22 which is depicting some examples of the actual data set. Again the geometric mean is computed to obtain the global effect. Finally we will sum-up all the reported cases by means of standard fuzzy arithmetic. Choosing for the sake of clarity the simplest common approach using standard triangular fuzzy numbers (TFN) to operationalize
the linguistic variables and geometric mean when multiplying TFN. In the same vein we will apply simple averaging to sum-up by classes the relevance of a set of cases.

The detailed list of all the considered events can be found on Appendix C the following table 6.6 expresses this analytic assessment in a processed and more condensed form. This table reports, also, on the Reliability subcharacteristic affected by each considered defect.

![Figure 6.6: Summary of Analysis Computations](image)

On that table, column "Defect" details those actually located in the project reports, followed by column "subCh" which reports on the Reliability subcharacteristic on which each defect impacts. This being accounted by column "f" frequency. Then column $\alpha_i$ is just the nor-
malized frequency and TFN the geometric mean of the “f” events as assessed following the metod explained on above. Then we found the normalization of such means necessary in order to allow comparisons.

In that way we have integrated the usage model and the defects and behavior analysis that we can use for further specific analysis. It has to be noted that the effect of the newly introduced defects is only implicitly captured by the user’s perception.

**Third Increment Results Analysis**

**Users vs Product assessment.** For a comparative analysis of the outcomes from the stakeholders evaluation according to its specific expectation in terms of system behaviour and the results from the assessment based in actual product maintenance project records, (from where we have got the data on identified defects) we will use the following tables. On table 6.23 we found the Stakeholders view and on table 6.24 the analytical outcomes.

<table>
<thead>
<tr>
<th></th>
<th>$T - T_0$</th>
<th>$\Delta$</th>
<th>CRISP</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>(5, 6.98, 9)-(4, 6.48, 8)</td>
<td>(1, 0.5, 1)</td>
<td>0.83</td>
</tr>
<tr>
<td>Rb</td>
<td>(5, 6.53, 9)-(4, 7.1, 10)</td>
<td>(1, -0.56, -1)</td>
<td>-0.19</td>
</tr>
<tr>
<td>Ft</td>
<td>(5, 6.39, 9)-(4, 6.54, 9)</td>
<td>(1, -0.15, 0)</td>
<td>0.28</td>
</tr>
<tr>
<td>Rc</td>
<td>(5, 6.40, 8)-(5, 7.34, 9)</td>
<td>(0, -0.94, -1)</td>
<td>-0.65</td>
</tr>
</tbody>
</table>

**Table 6.23:** FAHP Consensus on Reliability evolution

<table>
<thead>
<tr>
<th>subC</th>
<th>$\Sigma TFN$</th>
<th>Crisp</th>
<th>norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>(1.47206, 2.05338, 2.63431)</td>
<td>2.05325</td>
<td>0.42</td>
</tr>
<tr>
<td>Rb</td>
<td>(0.53726, 0.90615, 1.42579)</td>
<td>0.9564</td>
<td>0.19</td>
</tr>
<tr>
<td>Ft</td>
<td>(1.142, 1.63603, 2.32652)</td>
<td>1.701516667</td>
<td>0.35</td>
</tr>
<tr>
<td>Rc</td>
<td>(0.10661, 0.14354, 0.182)</td>
<td>0.14405</td>
<td>0.03</td>
</tr>
</tbody>
</table>

**Table 6.24:** Reliability Assessment from Defect Data

We easily found that both provide the same outcome in terms that the user assessment of the relative evolution of each sub-characteristic, from the better to the worst, $S > Ft > Rb > Rc$ is the very same that we found when assessing the impact for the fixed defects during the same time span and product releases, again $S > Ft > Rb > Rc$. This agreement constitutes an empirical validation of our proposed model and application method and, as a consequence, its suitability as a scientific tool in the analysis of the system Reliability.

We can, also, easily repeat the computations previously depicted in table 6.15 and 6.16 to compute the global perception of Reliability at this Case Study endpoint.
Now, as in precedent increment we compute Reliability through the proposed model as follows;

\[ R = W_{Ft} \times 6.16 + W_{Rc} \times 7 + W_{Rb} \times 6.51 + W_{St} \times 7.11 \]

\[ R = 0.1897 \times 6.79 + 0.1135 \times 6.47 + 0.5155 \times 6.84 + 0.1783 \times 6.993 = 6.795 \]

Again we got a fairly good agreement between the experts’ subjective assessment of 7.38 and the analytical outcome from our model 6.795 with a deviation of roughly 8%. It is to note that \( R_{T_0} < R_T \) that is, Reliability has slightly increased with the last major release. That’s good.

All the above also demonstrates, at least in great extent, our main hypothesis. That International Standards can be applied, within common industrial constraints, to determine the Reliability of a Software Driven System. To reinforce this conclusion we can show an application of this, now validated, proposal to a common problem in Software Industry, the one of decide whether a “bug” should be fixed now, later or … never.

### 6.3. A SOFTWARE MANAGEMENT PROBLEM

Among the main goals of Software Product Maintenance we can mention the one of promoting the Customer Satisfaction as well as the optimization of the spent resources in the process to pursue it. Customer Satisfaction is a very difficult to capture concept which in any case is tightly related to the user’s experience when using the product. Causes of stakeholder’s dissatisfaction are certainly, but not only, on the Software defects and its consequences in the form of system failures. That is, the Software Product Reliability.

From an industrial profit-oriented standpoint, thoroughly testing large and very large systems pursuing the fix of as much as possible defects is un-economics and most of times unnecessary. In that way, in most of the projects we could consider the role of the maintenance engineer...
will be on efficiently assuring the Product Quality avoiding such comprehensive testing. 
Unfortunately, knowing which defects affects the more (and the less) to user perception 
of Software Product Reliability is far from being an obvious task. On the following we are 
going to illustrate how we can address this issue by an exercise based on our model and 
application method.

Actually, what we are going to determine is which kind of Software Product Defects, accord-
ingly to our proposed taxonomy, have the higher impact on the Reliability of the system 
as perceived by their users. Then, is on fixing such bugs where we should put the money. 
Such is a valuable insight for technical managers constrained to work under budgetary 
constraints and the duty of maximize the positive impact of the maintenance activities.

In a way similar to the method applied to validate our model, the individual computation 
for each of the analyzed defects are grouped by classes accordingly to the taxonomy of 
reference as depicted. This is detailed in Appendix 3. Then its global effects on Reliability 
are summarized by means of a geometric mean. For comparison purposes the results are 
normalized taking as reference the number of user reports on each class and the total one, 
since it is the number of user’s reports which is of the interest for the group of stakeholders 
of reference for this case more than the raw number of events when in operation, which is 
implicitly captured by the Usage Model.

<table>
<thead>
<tr>
<th>Class</th>
<th>Defect type</th>
<th>Average geoTFN</th>
<th>CRISP</th>
<th>Normalized</th>
<th>CRISP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Miss Requirement</td>
<td>4</td>
<td>5.65</td>
<td>7.5</td>
<td>5.7</td>
</tr>
<tr>
<td>2</td>
<td>Miss or Wrong Function</td>
<td>4.67</td>
<td>6.12</td>
<td>7.67</td>
<td>6.145</td>
</tr>
<tr>
<td>3</td>
<td>Missed or wrong case</td>
<td>3.86</td>
<td>5.27</td>
<td>6.79</td>
<td>5.2975</td>
</tr>
<tr>
<td>4</td>
<td>Miss or wrong check</td>
<td>3.17</td>
<td>4.59</td>
<td>6.12</td>
<td>4.6175</td>
</tr>
<tr>
<td>5</td>
<td>Wrong logic - algorithm</td>
<td>2.63</td>
<td>4.3</td>
<td>6.31</td>
<td>4.385</td>
</tr>
<tr>
<td>6</td>
<td>Initialization</td>
<td>2.74</td>
<td>4.27</td>
<td>6</td>
<td>4.32</td>
</tr>
<tr>
<td>7</td>
<td>Synchronisation &amp; timer</td>
<td>2.5</td>
<td>4.16</td>
<td>6</td>
<td>4.205</td>
</tr>
<tr>
<td>8</td>
<td>Lack or wrong Mutex</td>
<td>3.1</td>
<td>4.7</td>
<td>6.55</td>
<td>4.7625</td>
</tr>
<tr>
<td>9</td>
<td>Error &amp; Exception handling</td>
<td>2.12</td>
<td>4.69</td>
<td>6.18</td>
<td>4.42</td>
</tr>
<tr>
<td>10</td>
<td>Data type handling</td>
<td>2.6</td>
<td>4</td>
<td>5.8</td>
<td>4.05</td>
</tr>
<tr>
<td>11</td>
<td>Coding error</td>
<td>2.93</td>
<td>4.57</td>
<td>6.4</td>
<td>4.6175</td>
</tr>
<tr>
<td>12</td>
<td>Wrong configuration</td>
<td>2</td>
<td>3.29</td>
<td>4.67</td>
<td>3.3125</td>
</tr>
</tbody>
</table>

Figure 6.7: Defect Classes Impact on Reliability
To enable a graphical expression of the results, which will ease their analysis in a decision making process, those fuzzy numbers have been "de-fuzzified" by means of the standard centroid method to obtain the "crisp" values as displayed on table 6.7 which summarize the performed computations. On the following fig 6.8 the normalized results are depicted in a way that we can clearly identify that the defects in the category 3 are the best candidate to focus the maintenance action in the objective of maximizing its impact regarding the stakeholders perception on the System Reliability while those efforts put in categories 12 and 7 are unlikely to result in a notable improvement of the user experience. This is a valuable insight to manage the allocated resources in a responsible and informed manner.

![Normalized Impact on Reliability](image)

**Figure 6.8: Normalized Impact on Reliability**

### 6.4. THE CASE STUDY: EMPIRICAL VALIDITY AND CONCLUSIONS

As a final section of our Case Study, and as it was explained in Chapter 2 we will analyze those potential threats to the validity of our study according to [104, 123] and how we addressed them in order to mitigate its undesired effects. Then we will draw several relevant conclusions.
6.4.1. Empirical Validity

The major threats to validity correspond to the main dimensions of the validity: Construct, Internal and External validity, but we also will consider Reliability.

**Construct Validity.** Refers to the aspect of validity that reflects to what extent the observations considered into the analysis represent what the researcher intend to investigate. In our analysis the construct validity is ensured by the high expertise profile of the expert committee members as well as the use of explicitly stated definitions for the concepts under study which largely contributes to a common understanding.

**Internal validity:** Due to the characteristic of the presented study internal validity threats do not apply. This aspect of validity is of concern when causal relations are examined, which is not the presented case. We do not consider that our relationship analysis from defect to behavior is enough as to claim causality but rather a path to gather relevant evidence in the pursuing of this goal.

**The external validity** is concerned with to what extent it is possible to generalize the findings. In the examined application the focus was in a particular class of stakeholders. Further generalizations would imply selecting experts from other contexts, for example, the system end-users or the organization’s board of directors. As this study was conducted on one single case it may limit its external validity. This is a generally recognized validity threat for the Case Study methodology \[104\]. As an alternative to generalizability, in this thesis we instead aim to minimize the threat to external validity by means of **transferability** \[104\], i.e. by providing a thorough case description allowing the reader to determine whether the results can be transferred to other contexts but above all we deem that our proposed model and application methodology is of ease application to any other software product in the aim to assess Reliability considered as a property with a relevant component of human perception.

**Reliability:** This aspect is concerned with to what extent the data and the analysis are dependent on the specific researchers. Hypothetically, if another researcher later on conducted the same study, the result should be the same. Threats to this aspect of validity is, for example, if it is not clear how to code collected data or if interview questions are unclear.
Our data collection protocol trend to ensure data consistency since rely on explicit definitions and reproducibility since explicitly stated as well as the processing technique, which is generally recognized as very performant when dealing with vagueness or even wrong data, promoting bias avoidance. Notwithstanding the fact that data from historical records was extracted and interpreted only by the main researcher have to raise a red flag regarding a possible bias. We consider this risk is largely mitigated by the use of an explicit taxonomy which we consider rather consistent and clear.

6.4.2. Main Conclusions from the Case Study

As mentioned, this work is conducted with the aim of analyze the suitability, in terms of efficiency and effectiveness of the SB-SRM approach in an industrial setting, with particular focus on the users’ perception on the delivered Quality and how can capture the stakeholders’ view on the topic. Industrial applicability is related not only to intrinsic attributes like soundness or accuracy but mainly with two essential extrinsic aspects; its ability to fit into the organization process with a minimum impact and minimal needs on resources (efficiency) and the capability to provide proper answer to users’ stated or implicit needs (effectiveness).

Also, at the beginnig of this section we have stated some particular objectives as:

- Gaining empirical evidence on the suitability of the proposed model and method.
- Shown that the analysis can be applied on a continous improvement process.
- Analysing the relationship between source code attributes and the exhibited behaviour.

It is from the experience on the Case Study application that we get the answer to these points. First of all, the model we have presented rely in a reduced number of parameters which are straightforward to obtain from simple linguistic assessment and by means of the well-known FAHP process as well as it is simple to capture stakeholders’ perception by means of the same process. None of these activities had demanded particular resources and the implication of each participant engineer was limited to several sort interviews and meetings. Providing their assessments was, itself, just matter of some minutes.
In terms of efficiency, the developed exercise has thus shown the ease of application without introducing overloads in the organization processes, although it suggests the use of *ad hoc* support tools in order to lighten even more this impact. The proposed model being conceptually simple and clear facilitates dealing with ambiguity and ease discussions about the definition of high level concepts by providing an agreement between the parties when using this framework as an analysis tool.

The ability of the model to estimate with satisfactory accuracy quantities needed by software managers, engineers, and users has been proved. First, in the second incremental step when assessing Reliability directly from experts’ assessment then by means of our model computation. Once again on the final incremental step when showing fairly good agreement between the two assessments, the one from subjective measures and the one from objective ones. The conclusion is then that in terms of effectiveness the presented analysis has shown a good agreement between the global perception as stated by the users and the outcome obtained by the proposed structural model and application framework. This agreement allow us to apply our model and method as an analysis tool.

As said, this framework design is intended to permit the empirical validation of the proposed SB-SRM model by means of a real-world application but also to provide a path for researching on the relationships between software product low-level attributes and the higher level Reliability Sub-characteristics.

We can, also, draw some conclusions on this topic. From data on figure 6.6 we can infer whether some particular kind of software defects are, in a reasonable extent, related to each of the sub-characteristics. Notably, in case of Stability we see that out of a total of 67 events 34 were due to missed or wrong case and still 13 more were related to issues regarding with requirements or functions. In any case, 47 of 67 are related with the functionality as designed and in a less extent as defined.

Regarding the Robustness and the Fault Tolerance, even if the data from our Case Study are not as drastically separated there are some clear trends. For the Robustness we see that 17 out of 42 are coding issues; only wrong logic and initialization issues. In an analogous manner, from 63 events impacting the Fault Tolerance, 36 are related to coding issues, all of them related with the Logic, Initialization, Exception handling and Threading synchronisation.
On the contrary just six events resulted related to Recoverability. Such a limited quantity forbid to infer any conclusion for this subcharacteristic.

The ultimate conclusion is that our proposed methodology permits the investigation of the relationship between the low-level software product attributes and the Reliability Sub-characteristics in a sound and efficient manner. In that way we have also provide some empirical evidence on the relationship between the software defects and the Reliability since showing how some classes of software defects impact on some characteristics and not in others. That is, our model can provide a qualitative answer to the question on the relationship between the low level attributes and the exhibited behavior since showing which defects affects each reliability sub-characteristic.

We conclude that this work provides, then, valuable empirical evidence on the correctness of the proposed model and the suitability of applying SB-SRM approach to capture the users’ perception on System Quality. Notwithstanding additional evidence is yet required.

Finally, and maybe the most important conclusion is that from the application experience on the Case Study we claim that our proposed model open a way to a broader application of the SB-SRM compatible with common industrial constraints. Our response to the identified obstacles has been validated and has shown that fit into Organization operations without introducing relevant disturbance but providing worth information for decision makers to manage the system in a disciplined manner.

All the above promotes the capability of this proposal to fit into the organization process with a minimum impact and minimal needs on resources but providing an appropriate descriptiveness and agreement with stakeholder’s expectation as to constituting a suitable engineering tool.
CHAPTER 7

RESEARCH OUTCOMES.

This chapter presents a summary of the contributions of this Thesis, an analysis of goal achievement and the results obtained. We also detail the publications where this research is published to date. The conclusions are also used as a basis to outline future lines of research.

What we know here is very little, but what we are ignorant of is immense.

Pierre-Simon Laplace

7.1. ANALYSIS OF RESEARCH GOALS

Software Quality is a multi-dimensional concept that is strongly influenced by the stakeholders’ perception and on which is commonly recognized that the Product Reliability is the central issue. Despite the huge body of knowledge regarding models and methods with the ultimate goal of improving or assessing Software Reliability there is no common agreement on what Software Reliability is, as a consequence different stakeholders use a variety of Software Reliability views.

In the aim to provide an alternative approach with which to integrate these several perspectives, this research relies on Software Quality International Standards as a basis to investigate how Software Reliability can be described and analysed in a way straightforward to apply in industrial context. In this Thesis a Software Reliability Model and an application methodology is thus proposed.

In such sense, the introductory chapter of this document states the following general objective: To develop a Standard-Based Model for Software Product Reliability description and
This general objective is then broken down into several specific objectives in order to manage the research activities. The fulfilment of each specific objective contributes toward achieving the general objective. In order to show how each specific objective was addressed, a summary of activities and results is therefore provided here below.

### Pertinence of the research and State-of-the-Art

1. O1. To perform a literature review of the software product reliability in order to produce a State-of-Art and identify trends in the research, proposed approaches and already identified issues.

   a) O1.1 To carry out a Systematic Mapping Study designed with the aim of identifying and categorizing a broad set of primary studies covering the work currently being considered by researchers as regards the various aspects of the reliability modelling of software driven systems, with a particular focus on Software Product Reliability.

   b) O1.2 To carry out a Systematic Literature Review with the aim of identify the current approaches to Software Reliability Modelling related to International Quality Standards and constitute the reference State of the Art for the present research

**Chapter 3: State-of-the-Art**, thoroughly examines the previous Literature on the topic to confirm that a gap in research exists on the topic of Software Reliability Models developed and applied following the guidelines of International Quality Standards. Moreover it allowed to identify the main impairments to a broad application in particular in real-world industrial context. First of all an SMS was performed from which the emergence of alternatives to the classical SRGM was confirmed and, as a consequence, a new Software Reliability Models taxonomy was proposed. An SLR was also conducted to mitigate the confirmed lack of research on SB-SRM. The main outcome of this SLR research is the formal identification of the main reasons for such lack of development as well as application in real-world industrial context. Both research are published in [41] and [43].
CHAPTER 7. RESEARCH OUTCOMES.

Issues and Fixes

1. O2. To identify the impairments for broader application of the Standard-Based Software Reliability Modelling (SB-SRM) into real-world industrial environment

2. O3. To define a software product reliability model to overcome such hindrances

Chapter 4: Analyse these identified issues which constitutes the main impairments and presents a strategy to overcome them developing a solution to the identified issues which is proposed together with an empirical validation strategy. The solution proposal is made up of a new innovative Software Reliability Model and an application methodology. The Model proposal is based on developing the latest normative advances in particular the ISO/IEC-SQuaRE series of International Standard on Software Quality. The main novelty of this model is that the relationship between the elements is derived from its conceptual hierarchy. That enables an analysis at different levels that can be mapped e.g to the different stakeholders’ expectations. The solution proposal presented in this chapter is already published (thus, verified) as part of [43] and once again in [42] together with some of the validation techniques applied in this thesis.

How to do it

1. O4. To develop an efficient framework that can be applied without introducing notable overload and thus promoting the SRM in daily industrial practice.

Chapter 5: Analyse and develop what is required to define an application method for our proposed model; notably introduce a detailed method for usage modelling and, at the best of our knowledge, an innovative method of application of the Case Study methodology as an Incremental Case Study method which fits particularly well into Industrial practice as a risk mitigation approach.

Both, the method and techniques to apply it are chosen and developed in the aim to promote their application within typical industrial context highly featured by the need of economic profit that is the efficiency on expended resources.
How it did it.

1. O5. Verification and Validation of the Proposal.

   a) O5.1 Verify the proposed solution correctness by means of formal peer reviews

   b) O5.2 Validate the proposed Model & framework by conducting Industrial Case Study aimed to analyse: Its Industrial applicability and its conceptual descriptiveness

Verification has occurred by means of the publication, after peer-review, of the research conducting to the hindrances identification, as well as the proposed solution to overcome them. All this is published in [41, 43] and [42] where, in addition, the two first steps of the validation strategy were also verified and presented to the Academic Community.

Chapter 6: Proves the rationality of the proposed model and suitability of the analysis method is by the application on a very large industrial system which provides empirical evidence on the conceptual descriptiveness capturing stakeholders’ views and industrial applicability in an efficient manner as well as on the relation between the low level software product attributes and the exhibited behaviour. This validation work is already partially published, two steps of three, in [42] as well it was presented at the 2017 IEEE International Conference on Software Quality, Reliability & Security.

In terms of effectiveness the presented analysis has shown a good agreement between the global perception as stated by the users and the outcome obtained by the proposed framework. We conclude that this constitutes valuable empirical evidence on the correctness of the proposed model, notwithstanding additional evidence is yet required.

In terms of efficiency, the developed exercise has shown the ease of application without introducing overloads in the organization processes, although it suggests the use of ad hoc support tools in order to lighten even more this impact. The model is conceptually simple and clear which facilitates dealing with ambiguity and discussion about the definition of high level concepts by providing an agreement between the parties when using this framework as an analysis tool.
All this promotes the capability of this proposal to fit into the organization processes with a minimum impact and minimal needs on resources but providing an appropriate descriptiveness and agreement with stakeholder’s expectation as to constituting a suitable engineering tool. This work provides, then, experimental evidence on the suitability of applying SB-SRM approach to assess Software Product Reliability in Industrial Environments in a sound, efficient and effective manner.

7.2. SUPPORT FOR RESULTS

As aforementioned, during the time this research was conducted several articles reporting on an important part of this Thesis were published in Software Engineering related journals and presented at conferences. The list of these articles is shown below:

International Journals.

  At 01-09-2019: Cited 28

  At 01-09-2019: Cited 22

International Conferences.

  At 01-09-2019: Cited 1

Total Citations. \((28 + 22 + 1) = 51\)
7.3. RESEARCH CONTRIBUTIONS

The research contributions of this Thesis can be categorized as follows:

1. We have built and published a complete State-of-the-Art as regards how International Standards are applied in Software Reliability Modeling

2. It is \textit{(at the best of our knowledge)} the first systematic review of SB-SRM literature
   
   \begin{itemize}
   \item[a)] We have coined the Standard-Based Software Reliability Modeling term and developed along the present dissertation what it is.
   \item[b)] We have also introduced a revised taxonomy for Software Reliability Modeling
   \item[c)] The SMS results have allowed us to identify a research gap on SB-SRM.
   \item[d)] The SLR results have allowed us to identify the main impairments to a broader application of SB-SRM in industrial environment.
   \end{itemize}

3. As a response, this work presents an innovative layout based on the conceptual hierarchy of its components and intended to model Software Reliability integrating the needs of the different stakeholders in a simple but highly descriptive manner.

4. Presents an innovative method for Case Study methodology application highly inspired in the incremental strategies for project risk mitigation.

5. Presents the \textit{(at the best of our knowledge)} first application of F-AHP to the analysis of Software Product Reliability.

6. This work provides empirical evidence on the capability of AHP and FAHP to deal with the human factor in Software Engineering.

7. Also contributes with empirical evidence on the relationship between the low level software product attributes and the exhibited behaviour.

8. The developed Case Study provides empirical evidence on the industrial suitability of our proposed model and application method.
7.4. FUTURE RESEARCH LINES.

Two main future research lines can be derived from the presented Thesis. On the one hand its extension to the more general concept of Dependability. On the other the exploration of alternatives regarding the aggregation technique.

The extension of the model and method to the wider concept of Dependability is deemed to be the natural path. Dependability is nowadays the mainstream concept integrating around Reliability the concepts of Maintainability and Security, often completed with Functionality and, why not, Sustainability.

To achieve this it would be required the extension of the proposed Taxonomy as well as its revision in order to assess whether the here applied description level is the more appropriate or not. This task is a real chalenge on itself.

A further research line is the integration of alternative aggregation techniques. Arriving to equivalent conclusions by means of different "measurement" instruments constitutes an important evidence of correctness. Beyond the application of more refined and rigorous Fuzzy computations other radically different techniques should be explored. Among the more exciting candidates we wonder on the suitability of Bayesian Networks.

In the long term, we deem that the presented model and application methodology constitutes a solid basis for the development of Rule Based Expert Systems which, in fine, would constitute the ideal industrial escenary. This hypothesis, obviously need of its own empirical development.
Appendices
This appendix reports on the details of the several searches strings and set-up we have carried out in order to gather the appropriate set of primary studies for our Systematic Literature Review.

Search in IEEE Digital Library

You searched for: (standard AND model AND software) AND (reliability OR maturity OR fault tolerance OR availability OR recoverability OR dependability)) You Refined by: Publication Year: 1991 - 2014 1.194 Results returned

Sorted by relevance, first 500 retained,

(((("software reliability" AND "ISO") OR ("software maturity" AND "ISO")) OR (("software availability" AND "ISO") OR ("software fault tolerance" AND "ISO"))) OR (("software recoverability" AND "ISO") OR ("software dependability" AND "ISO")))

67 Results returned

(((("software reliability" AND "ISO/IEC") OR ("software maturity" AND "ISO/IEC")) OR (("software availability" AND "ISO/IEC") OR ("software fault tolerance" AND "ISO/IEC"))) OR (("software recoverability" AND "ISO/IEC") OR ("software dependability" AND "ISO/IEC")))

19 Results returned

(((("software reliability" AND "square") OR ("software maturity" AND "square")) OR (("soft-
ware availability" AND "square") OR ("software fault tolerance" AND "square")) OR ("software recoverability" AND "square") OR ("software dependability" AND "square"))

37 Results returned

In a similar way, hits sorted by relevance;

("software reliability") AND model*) 2794 first 100 retained

("software reliability") AND characteristics) 314 first 100 retained

("software reliability") AND dimensions)

48 Results returned

("software reliability") AND features) 327 first 100 retained

("software reliability") AND components) 904 first 100 retained

("software reliability") AND factors) 498 first 100 retained

**Search in ACM Digital Library**

Sorted by relevance

Searching for: (((((((Title:"software reliability") and (Title:"ISO")) or ((Abstract:"software reliability") and (Abstract:"ISO"))) or (Keywords:"software reliability" AND Keywords:"ISO")))

or(((Title:"software maturity") and (Title:"ISO")) or ((Abstract:"software maturity") and (Abstract:"ISO"))) or (Keywords:"software maturity" AND Keywords:"ISO")))))

or(((Title:"software availability") and (Title:"ISO")) or ((Abstract:"software availability") and (Abstract:"ISO"))) or (Keywords:"software availability" AND Keywords:"ISO")))

or(((Title:"software fault tolerance") and (Title:"ISO")) or ((Abstract:"software fault tolerance") and (Abstract:"ISO"))) or (Keywords:"software fault tolerance" AND Keywords:"ISO")))

or(((Title:"software recoverability") and (Title:"ISO")) or ((Abstract:"software recoverability") and (Abstract:"ISO"))) or (Keywords:"software recoverability" AND Keywords:"ISO")))

or(((Title:"software dependability") and (Title:"ISO")) or ((Abstract:"software dependability") and (Abstract:"ISO")))
APPENDIX A. LITERATURE REVIEW SEARCH 161

or((Keywords:"software dependability" AND Keywords:"ISO")) and (not Publisher:IEEE) and (FtFlag:yes) and (AbstractFlag:yes)

Found 4

Searching for: (((((Title:"software reliability") and (Title:"ISO/IEC")) or ((Abstract:"software reliability") and (Abstract:"ISO/IEC"))) or((Keywords:"software reliability" AND Keywords:"ISO/IEC")) or(((Title:"software maturity") and (Title:"ISO/IEC")) or ((Abstract:"software maturity") and (Abstract:"ISO/IEC"))) or((Keywords:"software maturity" AND Keywords:"ISO/IEC")) or(((Title:"software availability") and (Title:"ISO/IEC")) or ((Abstract:"software availability") and (Abstract:"ISO/IEC"))) or((Keywords:"software availability" AND Keywords:"ISO/IEC")) or(((Title:"software fault tolerance") and (Title:"ISO/IEC")) or ((Abstract:"software fault tolerance") and (Abstract:"ISO/IEC"))) or((Keywords:"software fault tolerance" AND Keywords:"ISO/IEC")) or(((Title:"software recoverability") and (Title:"ISO/IEC")) or ((Abstract:"software recoverability") and (Abstract:"ISO/IEC"))) or((Keywords:"software recoverability" AND Keywords:"ISO/IEC"))) or(((Title:"software dependability") and (Title:"ISO/IEC")) or ((Abstract:"software dependability") and (Abstract:"ISO/IEC"))) or((Keywords:"software dependability" AND Keywords:"ISO/IEC")))

Found 1 duplicate one of the above

Searching for: (((((Title:"software reliability") and (Title:square)) or ((Abstract:"software reliability") and (Abstract:square)) or((Keywords:"software reliability" AND Keywords:square))) or(((Title:"software maturity") and (Title:square)) or ((Abstract:"software maturity") and (Abstract:square)) or((Keywords:"software maturity" AND Keywords:square))) or(((Title:"software availability") and (Title:square)) or ((Abstract:"software availability") and (Abstract:square)) or((Keywords:"software availability" AND Keywords:square))) or(((Title:"software fault tolerance") and (Title:square)) or ((Abstract:"software fault tolerance") and (Abstract:square)) or((Keywords:"software fault tolerance" AND Keywords:square))) or(((Title:"software recoverability") and (Title:square)) or ((Abstract:"software recoverability") and (Abstract:square)) or((Keywords:"software recoverability" AND Keywords:square))) or(((Title:"software dependability") and (Title:square)) or ((Abstract:"software dependability") and (Abstract:square)) or((Keywords:"software dependability" AND Keywords:square)))))

Found 1 duplicate one of the above
dependability" AND Keywords:square))) )and (not Publisher:IEEE) and (FtFlag:yes) and (AbstractFlag:yes) )

Found 2

Searching for: (Owner:GUIDE(not Publisher:IEEE) and (FtFlag:yes) and (AbstractFlag:yes))((((Title:standard) and (Title:model) and (Title:software)) and ((Title:reliability) or (Title:maturity) or (Title:"fault tolerance") or (Title:availability)or (Title:recoverability)or (Title:dependability)) )or(((Abstract:standard) and (Abstract:model) and (Abstract:software)) and ((Abstract:reliability) or (Abstract:maturity) or (Abstract:"fault tolerance") or (Abstract:availability)or (Abstract:recoverability)or (Abstract:dependability)) )or(((Keywords:standard) and (Keywords:model) and (Keywords:software)) and ((Keywords:reliability) or (Keywords:maturity) or (Keywords:"fault tolerance") or (Keywords:availability)or (Keywords:recoverability)or (Keywords:dependability))))

Found 95

Searching for: (Owner:GUIDE(not Publisher:IEEE) and (FtFlag:yes) and (AbstractFlag:yes))((Title:"software reliability") and (Title:dimension))or((Abstract:"software reliability") and (Abstract:dimension)) or ((Keywords:"software reliability") and (Keywords:dimension))

Found 42

Searching for: (Owner:GUIDE(not Publisher:IEEE) and (FtFlag:yes) and (AbstractFlag:yes))((Title:"software reliability") and (Title:features))or((Abstract:"software reliability") and (Abstract:features)) or ((Keywords:"software reliability") and (Keywords:features))

Found 42

Searching for: (Owner:GUIDE(not Publisher:IEEE) and (FtFlag:yes) and (AbstractFlag:yes))((Title:"software reliability") and (Title:characteristic))or((Abstract:"software reliability") and (Abstract:characteristic)) or ((Keywords:"software reliability") and (Keywords:characteristic))
APPENDIX A. LITERATURE REVIEW SEARCH

Found 59

for: (Owner:GUIDE(not Publisher:IEEE) and (FtFlag:yes) and (Abstract-Flag:yes) ((Title:"software reliability") and (Title:factor)) or ((Abstract:"software reliability") and (Abstract:factor)) or ((Keywords:"software reliability") and (Keywords:factor)))

Found 102

Searching for: (Owner:GUIDE(not Publisher:IEEE) and (FtFlag:yes) and (Abstract-Flag:yes) ((Title:"software reliability") and (Title:component)) or ((Abstract:"software reliability") and (Abstract:component)) or ((Keywords:"software reliability") and (Keywords:component)))

Found 159 first 100 retained

Searching for: (Owner:GUIDE(not Publisher:IEEE) and (FtFlag:yes) and (Abstract-Flag:yes) ((Title:"software reliability") and (Title:model*)) or ((Abstract:"software reliability") and (Abstract:model*)) or ((Keywords:"software reliability") and (Keywords:model*))

Found 685 first 100 retained

Search in Science Direct Digital Library

Sort by: Relevance - selected 1 articles found for: pub-date > 1990 and (tak(software reliability) AND tak(ISO)) OR (tak(software maturity) AND tak(ISO)) OR (tak(software availability) AND tak(ISO)) OR (tak(software fault tolerance) AND tak(ISO)) OR (tak(software recoverability) AND tak(ISO)) OR (tak(software dependability) AND tak(ISO)) [All Sources(Computer Science)]

1 articles found for: pub-date > 1990 and (tak(software reliability) AND tak(ISO/IEC)) OR (tak(software maturity) AND tak(ISO/IEC)) OR (tak(software availability) AND tak(ISO/IEC)) OR (tak(software fault tolerance) AND tak(ISO/IEC)) OR (tak(software recoverability) AND tak(ISO/IEC)) OR (tak(software dependability) AND tak(ISO/IEC)) [All Sources(Computer Science)]

3 articles found for: pub-date > 1990 and (tak(software reliability) AND tak(square)) OR
(tak(software maturity) AND tak(square)) OR (tak(software availability) AND tak(square)) OR (tak(software fault tolerance) AND tak(square)) OR (tak(software recoverability) AND tak(square)) OR (tak(software dependability) AND tak(square)) [All Sources(Computer Science)]

118 articles found for: pub-date > 1990 and tak(software reliability) AND tak(model*) [All Sources(Computer Science)]

11 articles found for: pub-date > 1990 and tak(software reliability) AND tak(characteristic) [All Sources(Computer Science)]

2 articles found for: pub-date > 1990 and tak(software reliability) AND tak(dimension) [All Sources(Computer Science)]

5 articles found for: pub-date > 1990 and tak(software reliability) AND tak(feature) [All Sources(Computer Science)]

20 articles found for: pub-date > 1990 and tak(software reliability) AND tak(component) [All Sources(Computer Science)]

24 articles found for: pub-date > 1990 and tak(software reliability) AND tak(factor) [All Sources(Computer Science)]

28 articles found for: pub-date > 1990 and (tak(standard) AND tak(model) AND tak(software)) AND (tak(reliability) OR tak(fault tolerance) OR tak(availability) OR tak(recoverability) OR tak(dependability)) [All Sources(Computer Science)]
APPENDIX B

SOFTWARE PRODUCT DEFECTS

TAXONOMY

On this appendix the *ad hoc* developed Taxonomy is detailed. Some remarks are needed at this stage. Our interest is only on those defects which have to do with the Reliability characteristics: Stability, Robustness, Fault Tolerance and Recoverability. This appendix describes, then over 40 Reliability-related bugs. The descriptions also include some examples when considered interesting information in order to clarify the concept.

- **Functionality**: Bugs having to do with the functionality as specified or as designed. Defects in this areas are mainly related to completeness and correctness issues. Completeness is about lack of required functionality; cases, features, functions, requirements or attributes which are not specified and therefore not implemented. Correctness includes; ambiguoness, requirement is inconsistent or incompatible with other requirements or with environment. We also cosidrer as wrong functionality when is correct as stated but it is not desirable or is not needed.

  - **As Specified**: Defects having to do with the product specification
    
    ○ Missed feature or requirement. Feature or requirement is required but has not been specified

    ○ Missed Function Function is required but has not been specified
○ Wrong feature or requirement: wrong or not required feature or requirement

○ Wrong function wrong or not required function

• As Designed: Defects having to do with the implementation/design of the specification. Specification itself is assumed to be correct.

○ Missed case: Case is required by the specification but no design has been provided

○ Missed check: Check is required by the specification but no design has been provided

○ Wrong case: wrong or not required case has been designed

○ Wrong check: Inadequate tests of user input, or passed parameter. Inadequate protection against corrupted data

• Data: Bugs in the definition, declaration, structure, or use of data. This category applies whether the object is declared statically in source code or created dynamically.

○ Definition: bugs in the definition, structure, scope and initialization of data: e.g., in DATA statements.

○ Wrong data structure: The scope, partition, or components to which the object applies is incorrectly specified. Duplicated definition of an object where allowed by the syntax. number of specified resources is insufficient or there is insufficient space (e.g., main memory, cache, registers, disc, etc.) to hold the declared resources.

○ Wrong data base schema: the data object type, as declared, is incorrect: e.g., integer instead of floating, short instead of long, pointer instead of integer, array instead of scalar, incorrect user-defined type.

○ Wrongly dimensioned data: for arrays and other objects which have a dimension (e.g., arrays, records, files) by which component objects can be
indexed, a bug in the dimension or in the minimum or maximum dimensions, or in redimensioning statements.

- **Contents misinterpretation**: a syntactically acceptable conflict between a local and/or global declaration of an object. Incorrect declaration, Local should be global, Global should be local.

  - **Handling**: Having to do with access and manipulation of data objects that are presumed to be correctly defined. Data are passed from one part of a program to another, and from one program to another. In the process, the data might be corrupted. In this context, accesses include read, write, modify, and (in some instances) create and destroy.

  - Incorrect data conversion: object undergoes incorrect type transformation: e.g., integer to floating, pointer to integer, specified type transformation is not allowed, required type transformation not done. Note, type transformation bugs can exist in any language, whether or not it is strongly typed, whether or not there are user-defined types.

  - Inconsistent data population: scaling or units (semantic) associated with object is incorrect, incorrectly transformed, or not transformed. initialization or default value of object is incorrect. Not to be confused with initialization and default bugs in declarations. This is a dynamic initialization bug. incorrect constant value for an object: e.g., a constant in an expression.

  - Data reference out of bounds: access to object is partly correct, but the object structure and its boundaries are handled incorrectly: e.g., fetching 8 characters of a string instead of 7, mishandling word boundaries, getting too much or too little of an object.

  - Data corruption: The data are stored on disk, tape, RAM, whatever. The process corrupts stored data by putting bad values into these files. Overwritten data, not saved data, discarded data

  - Extraneous output data: data not required is output.
**Coding**: Bugs specifically related component’s structure: i.e., the implementation of the software, thus the source code.

- **Logic and control flow.** Defects specifically related to the control flow of the program or the order and extent to which things are done, as distinct from what is done. The control flow of a program describes what it will do next, under what circumstances. A control flow error occurs when the program does the wrong thing next
  
  - Wrong logic or algorithm: this defect refers to any issue directly related to control flow predicates like case selections, illogical or impossible conditions. bugs having to do with the control of loops, inclusion infinite loop and wrong criteria for ending loops.
  
  - Path left out: Code for which there is no combination of input values which will cause that code to be executed.
  
  - Dead-end code: code segments which once entered cannot be exited, even though it was intended that an exit be possible. Both, conditional statements and loops can be nested
  
  - Improper nesting: Both, conditional statements and loops can be nested (completely included) inside another. It is not possible (without error) to start inside another but to end outside of it.
  
  - Duplicate or unnecessary processing.

- **Processing** Bugs related to processing under the assumption that the control flow is correct.

  - Wrong initialization: Defects related to the program initializing to the wrong state or failing to initialize. bugs in initialization of variables, expressions, functions, etc. used in processing, excluding initialization bugs associated with declarations and data statements and loop initialization.
○ **Arithmetic**: bugs related to evaluation of arithmetic expressions. Wrong arithmetic operator or function used, syntactically correct bug in placement of parentheses or other arithmetic delimiters error in use of sign. bugs in symbolic processing of algebraic expressions.

○ **String wrongly handled**: bug in string manipulation. the beginning or the tail of a string is cut off when it should not have been or not cut off when it should be. Strings are concatenated in wrong order or concatenated when they should not be.

○ **Wrong format**: insufficient or excessive precision, insufficient accuracy and other bugs related to number representation system used.

○ **Exception or error not handled**: Exception conditions such as illogicals, resource problems, failure modes, which require special handling, are not correctly handled or the wrong exception-handling mechanisms are used. Error handling errors include failure to anticipate the possibility of errors and protect against them, failure to notice error conditions, and failure to deal with detected errors in a reasonable way.

○ **Boundary condition**: The program may use the wrong starting or ending address of a set of data. the values or expressions which define a domain boundary are wrong: e.g., "X>=6" instead of "X>=3." X>=0" instead of "X>0." Or "IF X>0 AND Y>0 .." instead of "IF X>0 OR Y>0..."

○ **Typo and clerical error**: Spelling mistakes, e.g. in messages displayed to the user.

- **Integration**: Bugs having to do with the integration of, and interfaces between, components. The components themselves are assumed to be correct.

- **Internal Interfaces**: Defects related to the interfaces between communicating components with the program under test. The components itself are assumed to be correct. In this context, direct or indirect transfer of data or control information via a memory object such as tables, dynamically allocated resources, or files,
constitute an internal interface.

- Invalid input. invoked component not initialized or initialized to the wrong state or with incorrect data. the place or state in the invoking component at which the invoked component was invoked is wrong.

- Wrong library. Defects related with the use of dynamically loaded libraries. More in general, bugs having to do with how software components are invoked. In this sense, a "component" can be a subroutine, function, macro, program, program segment, or any other sensible processing component

- Lack of mutex; multi-threading problems. A Mutex is a mutually exclusive flag. It acts as a gate keeper to a section of code allowing one thread in and blocking access to all others. that lead to race condition which is a fault. We include here semaphores although not the same yet similar.

- Wrong mutex: (Interlock bug) where objects are in simultaneous use by more than one caller, interlocks and synchronization mechanisms may be used to assure that all data are current and changed by only one caller at a time. These are not bugs in the interlock or synchronization mechanism but in the use of that mechanism.

- Wrong message or parameters. having to do with the parameters of the invocation, their number, order, type, location, values, etc. parameters of the invocation are incorrectly specified, incorrect invocation parameter type used or structural details of parameter used are incorrect: e.g., size, number of fields, subtypes. Parameter values are incorrect or are in the wrong order, too many parameters, too few parameters.

- **External Interfaces.** Defects having to do with external interfaces, such as I/O devices and/or drivers, or other software not operating under the same control structure.

- Wrong I/O protocol: The communication protocol between the computer and a device or between two computers specifies such things as when the
computer can send data, at what speed, and with what characteristics (parity, stop bits, etc.). A device might send data or respond out of turn, or it might send the data in the wrong format. Also bugs related to incorrect interrupt handling or setting up for interrupts: e.g., wrong handler invoked, failure to block or unblock interrupts.

- Assumptions on external I/O: The program tries to use a new device or store more data in memory, but can’t. The following conditions should be self-explanatory: Full disk, Full disk directory, Full memory area, Full print queue, Full message queue, Full stack, Disk not in drive, Disk drive out of service, No disk drive, Printer off line, Printer out of paper

- Wrong resource used: For example, the program prints data on the screen instead of the printer.

- Resource deadlock. External resources are also shared thus the program can fall in a deadlock if any resource end up locked by a third subsystem. Includes bugs having to do with timings and data rates for external devices such as: not meeting specified timing requirements (too long or too short), forcing too much throughput, not accepting incoming data rates.

- Devices error handling. Ignores hardware fault or error conditions or inadequate protection against operating system bugs. The program should assume that devices it can connect to will fail. Many devices can send back messages (set bits) that warn that something is wrong. If one does, the program should stop trying to interact with it and should report the problem to a human or to a higher level control program. The operating system has bugs. Application programs can trigger some of them. If the application programmer knows, for example, that the system will crash if he sends data to the printer too soon after sending it to the disk drive, he should make sure that his program can’t do that under any circumstances.

- **Documentation**: Bugs having to do with the System Operation Documents since having to do on how the Software product is used (*thus related to its Reliability*) both,
in nominal operation and (more important) during recovery from failure operations.

- Lack of Recovery procedure: If no procedure exists to recover the system from a particular failure the down-time will dramatically increase. This clearly impact the system availability, thus Reliability.

- Wrong Recovery procedure: When the procedure exists but due to defects on it the recovery time is greater than necessary.

- Wrong Operation procedure: If a defective operation procedure induces the end-user to make a mistake its perception of how Reliable the system is will be degraded. Also wrong operations are prone to unveil latent defects on the software then originating failures.
The following table displays the assessment of each of the product defects retained for the Reliability analysis. Among the reviewed project reports there were also a certain number of them unrelated to Reliability. Those were discarded as out-of-scope. Defects have been grouped by classes in order to ease the analysis and the dissemination of the results.

The linguistic scale on this assessment correspond to a classic nine points scale: Extreme, Severe, Relevant, Little, Least considering intermediate values. The global assessment is aggregated by means of the average and geometric mean as it has been explained on Chapter 5.

Under the "Class" labeling we have grouped defects of close nature as described on the legend on green background. This to enhance the legibility of the raw data. The column "UM" corresponds to the Usage Model classification and the "manifested misbehavior" column reports on the failure as perceived by an end user. This data is of paramount importance to determine both the visibility and the final relevance of each of the considered events.
<table>
<thead>
<tr>
<th>Class</th>
<th>UM Relevancy</th>
<th>Manifested misBehavior</th>
<th>visibility</th>
<th>geo-TFN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missed Requirement</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ga+</td>
<td>Relevant</td>
<td>Perturbed oper</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Aa-</td>
<td>Severe</td>
<td>Absent feature</td>
<td>Severe</td>
<td>6</td>
</tr>
<tr>
<td>Bb-</td>
<td>Relevant</td>
<td>Loss funct</td>
<td>Severe</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>Severe</td>
<td>absent Feature</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Gb++</td>
<td>Severe</td>
<td>Loss func</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Cb</td>
<td>Little</td>
<td>Wrong oper</td>
<td>Relevant</td>
<td>2</td>
</tr>
<tr>
<td>Gb+++</td>
<td>Severe</td>
<td>Wrong oper</td>
<td>Severe</td>
<td>6</td>
</tr>
<tr>
<td>Cb++</td>
<td>Severe</td>
<td>Wrong oper</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Missed or Wrong Function</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ca++</td>
<td>Extreme</td>
<td>Wrong oper</td>
<td>Severe</td>
<td>6</td>
</tr>
<tr>
<td>Bb-</td>
<td>Relevant</td>
<td>Feature lost</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Gb++</td>
<td>Relevant</td>
<td>Perturbed operation</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Bc+</td>
<td>Severe</td>
<td>Wrong info</td>
<td>Severe</td>
<td>6</td>
</tr>
<tr>
<td>Ea</td>
<td>Relevant</td>
<td>Func lost</td>
<td>Severe</td>
<td>4</td>
</tr>
<tr>
<td>Ab++</td>
<td>Severe</td>
<td>Lost funct</td>
<td>Severe</td>
<td>6</td>
</tr>
<tr>
<td>Missed or wrong case</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ca-</td>
<td>Little</td>
<td>Transaction fails</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Ab+</td>
<td>Relevant</td>
<td>Loss function</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Aa--</td>
<td>Relevant</td>
<td>App stuck</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Ga+</td>
<td>Relevant</td>
<td>function lost</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Da++</td>
<td>Relevant</td>
<td>Lost fuction</td>
<td>Severe</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>Extreme</td>
<td>System lost</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Da+</td>
<td>Relevant</td>
<td>Process crash</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Bc-</td>
<td>Little</td>
<td>Wrong info</td>
<td>Relevant</td>
<td>2</td>
</tr>
<tr>
<td>Da+</td>
<td>Little</td>
<td>Operation fails</td>
<td>Relevant</td>
<td>2</td>
</tr>
<tr>
<td>Ea</td>
<td>Relevant</td>
<td>Loss Function</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Da++</td>
<td>Relevant</td>
<td>Abort transact</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Ga+</td>
<td>Relevant</td>
<td>Perturbed oper</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Ga++</td>
<td>Severe</td>
<td>Funct lost</td>
<td>Severe</td>
<td>6</td>
</tr>
<tr>
<td>Aa---</td>
<td>Little</td>
<td>Wrong info</td>
<td>Relevant</td>
<td>2</td>
</tr>
<tr>
<td>Cb+</td>
<td>Relevant</td>
<td>Oper fails</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Cb+</td>
<td>Relevant</td>
<td>Oper fails</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Bb-</td>
<td>Relevant</td>
<td>Oper fails</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Ca</td>
<td>Relevant</td>
<td>absent funct</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Ba-</td>
<td>Relevant</td>
<td>absent funct</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Bc</td>
<td>Relevant</td>
<td>Wrong info</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Bc</td>
<td>Relevant</td>
<td>Absent funct</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Bc+</td>
<td>Severe</td>
<td>Feature lost</td>
<td>Severe</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Function</td>
<td>Severity</td>
<td>Problem</td>
<td>Relevancy</td>
</tr>
<tr>
<td>----</td>
<td>---------------</td>
<td>----------</td>
<td>---------</td>
<td>-----------</td>
</tr>
<tr>
<td>Ea+</td>
<td>Severe</td>
<td></td>
<td>Function lost</td>
<td>Relevant</td>
</tr>
<tr>
<td>Da++</td>
<td>Relevant</td>
<td></td>
<td>Wrong info</td>
<td>Severe</td>
</tr>
<tr>
<td>Gb++</td>
<td>Relevant</td>
<td></td>
<td>Wrong info</td>
<td>Severe</td>
</tr>
<tr>
<td>Ca+</td>
<td>Severe</td>
<td></td>
<td>Wrong oper</td>
<td>Severe</td>
</tr>
<tr>
<td>Gc++</td>
<td>Relevant</td>
<td></td>
<td>Lost func</td>
<td>Severe</td>
</tr>
<tr>
<td>Gb++</td>
<td>Relevant</td>
<td></td>
<td>Wrong alarms</td>
<td>Relevant</td>
</tr>
<tr>
<td>Gb+</td>
<td>Little</td>
<td></td>
<td>Wrong alarms</td>
<td>Relevant</td>
</tr>
<tr>
<td>Da++</td>
<td>Relevant</td>
<td></td>
<td>Wrong oper</td>
<td>Relevant</td>
</tr>
<tr>
<td>Ca+</td>
<td>Severe</td>
<td></td>
<td>Wrong oper</td>
<td>Relevant</td>
</tr>
<tr>
<td>Ca+</td>
<td>Severe</td>
<td></td>
<td>Wrong oper</td>
<td>Severe</td>
</tr>
<tr>
<td>Ea</td>
<td>Relevant</td>
<td></td>
<td>Lost funct</td>
<td>Relevant</td>
</tr>
<tr>
<td>Ca</td>
<td>Relevant</td>
<td></td>
<td>Wrong oper</td>
<td>Relevant</td>
</tr>
<tr>
<td>Aa-</td>
<td>Severe</td>
<td></td>
<td>Wrong oper</td>
<td>Relevant</td>
</tr>
<tr>
<td>Aa-</td>
<td>Severe</td>
<td></td>
<td>Wrong info</td>
<td>Severe</td>
</tr>
<tr>
<td>Bc</td>
<td>Relevant</td>
<td></td>
<td>Alarms flood</td>
<td>Relevant</td>
</tr>
<tr>
<td>Ga++</td>
<td>Relevant</td>
<td></td>
<td>Wrong info</td>
<td>Relevant</td>
</tr>
<tr>
<td>Ga++</td>
<td>Relevant</td>
<td></td>
<td>Wrong info</td>
<td>Relevant</td>
</tr>
<tr>
<td>Ca+</td>
<td>Severe</td>
<td></td>
<td>Lost func</td>
<td>Relevant</td>
</tr>
<tr>
<td>Ga++</td>
<td>Severe</td>
<td></td>
<td>Wrong oper</td>
<td>Severe</td>
</tr>
<tr>
<td>Ga+</td>
<td>Relevant</td>
<td></td>
<td>Funct lost</td>
<td>Relevant</td>
</tr>
</tbody>
</table>

| Miss or wrong check | 3.17 | 4.59 | 6.12 |

<table>
<thead>
<tr>
<th>Class</th>
<th>UM</th>
<th>Relevancy</th>
<th>Manifed misBehavior</th>
<th>visibility</th>
<th>geo-TFN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Da++</td>
<td>Relevant</td>
<td>Several Crash</td>
<td>Relevant</td>
<td>4</td>
<td>5.0</td>
</tr>
<tr>
<td>Aa--</td>
<td>Relevant</td>
<td>Service Crash</td>
<td>Relevant</td>
<td>4</td>
<td>5.0</td>
</tr>
<tr>
<td>Bb-</td>
<td>Relevant</td>
<td>Loss funct</td>
<td>Little</td>
<td>2</td>
<td>3.87</td>
</tr>
<tr>
<td>Ca</td>
<td>Relevant</td>
<td>Wrong exec</td>
<td>Little</td>
<td>2</td>
<td>3.87</td>
</tr>
<tr>
<td>Ea</td>
<td>Relevant</td>
<td>App Freeze</td>
<td>Little</td>
<td>2</td>
<td>3.87</td>
</tr>
<tr>
<td>Ga+</td>
<td>Relevant</td>
<td>Service Crash</td>
<td>Little</td>
<td>2</td>
<td>3.87</td>
</tr>
<tr>
<td>Da++</td>
<td>Relevant</td>
<td>Process Crash</td>
<td>Relevant</td>
<td>4</td>
<td>5.0</td>
</tr>
<tr>
<td>Ca</td>
<td>Relevant</td>
<td>Operational err</td>
<td>Relevant</td>
<td>4</td>
<td>5.0</td>
</tr>
<tr>
<td>Ga+</td>
<td>Relevant</td>
<td>Process Crash</td>
<td>Relevant</td>
<td>4</td>
<td>5.92</td>
</tr>
<tr>
<td>Ea</td>
<td>Relevant</td>
<td>Funct Lost</td>
<td>Severe</td>
<td>4</td>
<td>5.92</td>
</tr>
<tr>
<td>Aa-</td>
<td>Severe</td>
<td>Operat Error</td>
<td>Severe</td>
<td>6</td>
<td>7.0</td>
</tr>
<tr>
<td>Aa-</td>
<td>Severe</td>
<td>Operat Error</td>
<td>Severe</td>
<td>6</td>
<td>7.0</td>
</tr>
<tr>
<td>Bb-</td>
<td>Relevant</td>
<td>Wrong info</td>
<td>Relevant</td>
<td>4</td>
<td>5.0</td>
</tr>
<tr>
<td>Ga-</td>
<td>Least</td>
<td>Not observed</td>
<td>Least</td>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>Aa--</td>
<td>Relevant</td>
<td>Loss funtion</td>
<td>Little</td>
<td>2</td>
<td>3.87</td>
</tr>
</tbody>
</table>

Wrong logic – algorithm | 2.63 | 4.30 | 6.31 |

<table>
<thead>
<tr>
<th>Class</th>
<th>UM</th>
<th>Relevancy</th>
<th>Manifed misBehavior</th>
<th>visibility</th>
<th>geo-TFN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gc</td>
<td>Least</td>
<td>Poor perform</td>
<td>Relevant</td>
<td>0</td>
<td>2.23</td>
</tr>
<tr>
<td>Ca</td>
<td>Relevant</td>
<td>stop exec</td>
<td>Little</td>
<td>2</td>
<td>3.87</td>
</tr>
<tr>
<td>Class</td>
<td>UM</td>
<td>Relevancy</td>
<td>Manifested misBehavior</td>
<td>visibility</td>
<td>geo-TFN</td>
</tr>
<tr>
<td>-------</td>
<td>----</td>
<td>-----------</td>
<td>-------------------------</td>
<td>------------</td>
<td>---------</td>
</tr>
<tr>
<td>Ga+</td>
<td>Relevant</td>
<td>Loss function</td>
<td>Little</td>
<td>2</td>
<td>3.87</td>
</tr>
<tr>
<td>Gb+</td>
<td>Little</td>
<td>Lost Function</td>
<td>Little</td>
<td>2</td>
<td>3.0</td>
</tr>
<tr>
<td>Aa--</td>
<td>Relevant</td>
<td>“Troubles”</td>
<td>Little</td>
<td>2</td>
<td>3.87</td>
</tr>
<tr>
<td>Gb</td>
<td>Least</td>
<td>Wrong Info</td>
<td>Little</td>
<td>0</td>
<td>1.73</td>
</tr>
<tr>
<td>Gb+</td>
<td>Little</td>
<td>Wrong info</td>
<td>Little</td>
<td>2</td>
<td>3.0</td>
</tr>
<tr>
<td>Bb</td>
<td>Severe</td>
<td>Wrong data used</td>
<td>Severe</td>
<td>6</td>
<td>7.0</td>
</tr>
<tr>
<td>Ca+</td>
<td>Severe</td>
<td>Loss funct</td>
<td>Little</td>
<td>2</td>
<td>4.58</td>
</tr>
<tr>
<td>Ga+</td>
<td>Relevant</td>
<td>Wrong oper</td>
<td>Little</td>
<td>2</td>
<td>3.87</td>
</tr>
<tr>
<td>Aa--</td>
<td>Relevant</td>
<td>User-app crash</td>
<td>Severe</td>
<td>4</td>
<td>5.92</td>
</tr>
<tr>
<td>Ea</td>
<td>Relevant</td>
<td>Alarms flood</td>
<td>Relevant</td>
<td>4</td>
<td>5.0</td>
</tr>
<tr>
<td>Bc</td>
<td>Relevant</td>
<td>Wrong info</td>
<td>Little</td>
<td>2</td>
<td>3.87</td>
</tr>
<tr>
<td>Bb--</td>
<td>Little</td>
<td>Process Crash</td>
<td>Relevant</td>
<td>2</td>
<td>3.87</td>
</tr>
<tr>
<td>Bc</td>
<td>Relevant</td>
<td>Wrong info</td>
<td>Little</td>
<td>2</td>
<td>3.87</td>
</tr>
<tr>
<td>Aa--</td>
<td>Little</td>
<td>Wrong alarm</td>
<td>Severe</td>
<td>2</td>
<td>4.58</td>
</tr>
<tr>
<td>Bb</td>
<td>Severe</td>
<td>Wrong oper</td>
<td>Severe</td>
<td>6</td>
<td>7.0</td>
</tr>
<tr>
<td>Aa--</td>
<td>Severe</td>
<td>Transact fails</td>
<td>Relevant</td>
<td>4</td>
<td>5.92</td>
</tr>
<tr>
<td>Ca++</td>
<td>Severe</td>
<td>Lost feature</td>
<td>Relevant</td>
<td>4</td>
<td>5.92</td>
</tr>
<tr>
<td>Ga+</td>
<td>Relevant</td>
<td>Impos to start</td>
<td>Little</td>
<td>2</td>
<td>3.87</td>
</tr>
<tr>
<td>Ga+</td>
<td>Relevant</td>
<td>Function lost</td>
<td>Relevant</td>
<td>4</td>
<td>5.0</td>
</tr>
<tr>
<td>Ga+</td>
<td>Relevant</td>
<td>Lost feature</td>
<td>Relevant</td>
<td>4</td>
<td>5.0</td>
</tr>
<tr>
<td>Bb-</td>
<td>Relevant</td>
<td>Wrong info</td>
<td>Little</td>
<td>2</td>
<td>3.87</td>
</tr>
<tr>
<td>Ba-</td>
<td>Relevant</td>
<td>Wrong info</td>
<td>Relevant</td>
<td>4</td>
<td>5.0</td>
</tr>
</tbody>
</table>

Initialization 2.74 4.27 6
<table>
<thead>
<tr>
<th>Class</th>
<th>Relevancy</th>
<th>Manifested misBehavior</th>
<th>visibility</th>
<th>geo-TFN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gb+</td>
<td>Little</td>
<td>Wrong Info</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Gb++</td>
<td>Relevant</td>
<td>Wrong exec</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Gb+++</td>
<td>Severe</td>
<td>Funct lost</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Da++</td>
<td>Relevant</td>
<td>Lost Function</td>
<td>Little</td>
<td>2</td>
</tr>
</tbody>
</table>

**Synchronization & Timer**  
(Lack of: synchro mech, timer, interface, message pass …)

<table>
<thead>
<tr>
<th>Class</th>
<th>Relevancy</th>
<th>Manifested misBehavior</th>
<th>visibility</th>
<th>geo-TFN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bb-</td>
<td>Relevant</td>
<td>Function hung</td>
<td>Little</td>
<td>1.73</td>
</tr>
<tr>
<td>Bb-</td>
<td>Relevant</td>
<td>Process Crash</td>
<td>Little</td>
<td>1.73</td>
</tr>
<tr>
<td>Ca</td>
<td>Relevant</td>
<td>Process Crash</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Ga</td>
<td>Little</td>
<td>Service Crash</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Bb</td>
<td>Severe</td>
<td>Process Crash</td>
<td>Little</td>
<td>4.58</td>
</tr>
<tr>
<td>Bb</td>
<td>Severe</td>
<td>Process Crash</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Bb-</td>
<td>Relevant</td>
<td>Process Crash</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Ea+</td>
<td>Severe</td>
<td>Loss function</td>
<td>Little</td>
<td>4</td>
</tr>
<tr>
<td>Ea</td>
<td>Relevant</td>
<td>Wrong oper</td>
<td>Severe</td>
<td>4</td>
</tr>
<tr>
<td>Ca</td>
<td>Relevant</td>
<td>Wrong oper</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Ga+</td>
<td>Relevant</td>
<td>service crash</td>
<td>Relevant</td>
<td>4</td>
</tr>
</tbody>
</table>

**Error & Exception handling**

<table>
<thead>
<tr>
<th>Class</th>
<th>Relevancy</th>
<th>Manifested misBehavior</th>
<th>visibility</th>
<th>geo-TFN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gb+</td>
<td>Little</td>
<td>Wrong info</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Ab</td>
<td>Little</td>
<td>Loss Funct</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Bb-</td>
<td>Relevant</td>
<td>Wrong info</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Gc+</td>
<td>Little</td>
<td>Wrong info</td>
<td>Relevant</td>
<td>2</td>
</tr>
<tr>
<td>Bc</td>
<td>Relevant</td>
<td>Process Hung</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Bc</td>
<td>Relevant</td>
<td>Loss Function</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Bb-</td>
<td>Relevant</td>
<td>Process crash</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Ea</td>
<td>Relevant</td>
<td>Loss function</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Bc+</td>
<td>Severe</td>
<td>LostFunction</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Bb</td>
<td>Severe</td>
<td>App Hung</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Aa-</td>
<td>Severe</td>
<td>Process Crash</td>
<td>Relevant</td>
<td>4</td>
</tr>
</tbody>
</table>

**Data type handling**  
(string, formats …)

<table>
<thead>
<tr>
<th>Class</th>
<th>Relevancy</th>
<th>Manifested misBehavior</th>
<th>visibility</th>
<th>geo-TFN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gb</td>
<td>Least</td>
<td>Wrong output</td>
<td>Little</td>
<td>0</td>
</tr>
<tr>
<td>Gb</td>
<td>Least</td>
<td>imperffunction</td>
<td>Little</td>
<td>0</td>
</tr>
<tr>
<td>Cb+</td>
<td>Relevant</td>
<td>Loss of data</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Class</td>
<td>UM</td>
<td>Relevancy</td>
<td>Manifested misBehavior</td>
<td>visibility</td>
</tr>
<tr>
<td>-------</td>
<td>----</td>
<td>-----------</td>
<td>-------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>Ca</td>
<td>Relevant</td>
<td>User-App Hung</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Aa--</td>
<td>Relevant</td>
<td>User-App crash</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Aa--</td>
<td>Relevant</td>
<td>Process crash</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Ea</td>
<td>Relevant</td>
<td>Process crash</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Bc</td>
<td>Relevant</td>
<td>Wrong info</td>
<td>Little</td>
<td>2</td>
</tr>
<tr>
<td>Aa--</td>
<td>Relevant</td>
<td>Wrong info</td>
<td>Relevant</td>
<td>4</td>
</tr>
<tr>
<td>Bc-</td>
<td>Little</td>
<td>Wrong info</td>
<td>Relevant</td>
<td>2</td>
</tr>
</tbody>
</table>

**Coding error**

<table>
<thead>
<tr>
<th>Class</th>
<th>UM</th>
<th>Relevancy</th>
<th>Manifested misBehavior</th>
<th>visibility</th>
<th>geo-TFN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ga</td>
<td>Little</td>
<td>Function lost</td>
<td>Little</td>
<td>2</td>
<td>3.0</td>
</tr>
<tr>
<td>Bc</td>
<td>Relevant</td>
<td>Poor Performane</td>
<td>Little</td>
<td>2</td>
<td>3.87</td>
</tr>
<tr>
<td>Ca</td>
<td>Little</td>
<td>Service block</td>
<td>Little</td>
<td>2</td>
<td>3.0</td>
</tr>
<tr>
<td>Da++</td>
<td>Relevant</td>
<td>Operation fails</td>
<td>Relevant</td>
<td>4</td>
<td>5.0</td>
</tr>
<tr>
<td>Fb</td>
<td>Little</td>
<td>Wrong info</td>
<td>Relevant</td>
<td>2</td>
<td>3.87</td>
</tr>
<tr>
<td>Bc</td>
<td>Relevant</td>
<td>Views Crash</td>
<td>Relevant</td>
<td>4</td>
<td>5.0</td>
</tr>
<tr>
<td>Bb</td>
<td>Severe</td>
<td>Alarms flood</td>
<td>Little</td>
<td>2</td>
<td>4.58</td>
</tr>
<tr>
<td>Ca</td>
<td>Relevant</td>
<td>Loss function</td>
<td>Relevant</td>
<td>4</td>
<td>5.0</td>
</tr>
<tr>
<td>Ea-</td>
<td>Little</td>
<td>Wrong info</td>
<td>Relevant</td>
<td>2</td>
<td>3.87</td>
</tr>
<tr>
<td>Bb-</td>
<td>Relevant</td>
<td>Alarma flood</td>
<td>Severe</td>
<td>4</td>
<td>5.92</td>
</tr>
<tr>
<td>Ga++</td>
<td>Severe</td>
<td>Oper fails</td>
<td>Relevant</td>
<td>4</td>
<td>5.92</td>
</tr>
<tr>
<td>Ea</td>
<td>Relevant</td>
<td>Wrong oper</td>
<td>Little</td>
<td>2</td>
<td>3.87</td>
</tr>
<tr>
<td>Ga++</td>
<td>Severe</td>
<td>Feature lost</td>
<td>Relevant</td>
<td>4</td>
<td>5.92</td>
</tr>
<tr>
<td>Eb+</td>
<td>Severe</td>
<td>Wrong oper</td>
<td>Relevant</td>
<td>4</td>
<td>5.92</td>
</tr>
<tr>
<td>Ca</td>
<td>Relevant</td>
<td>Wrong info</td>
<td>Little</td>
<td>2</td>
<td>3.87</td>
</tr>
</tbody>
</table>

**Wrong config**

<table>
<thead>
<tr>
<th>Class</th>
<th>UM</th>
<th>Relevancy</th>
<th>Manifested misBehavior</th>
<th>visibility</th>
<th>geo-TFN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Relevant</td>
<td>Funct lost</td>
<td>Relevant</td>
<td>2</td>
<td>3.0</td>
</tr>
<tr>
<td>Bb</td>
<td>Severe</td>
<td>Wrong alarms</td>
<td>Little</td>
<td>2</td>
<td>3.87</td>
</tr>
<tr>
<td>Ea-</td>
<td>Little</td>
<td>Perturbed use</td>
<td>Little</td>
<td>2</td>
<td>3.0</td>
</tr>
</tbody>
</table>


