QRMF: A Multi-Perspective Framework for Quality Requirements Modelling

A thesis submitted to the University of Manchester for the degree of Doctor of Philosophy in the Faculty of Engineering and Physical Sciences

2014

Kawther Abdulelah Saeedi

School of Computer Science
Table Of Contents

TABLE OF CONTENTS .................................................................................................................. 2

LIST OF FIGURES ..................................................................................................................... 7

LIST OF TABLES .......................................................................................................................... 9

ABSTRACT .................................................................................................................................. 10

DECLARATION ........................................................................................................................... 11

COPYRIGHT STATEMENT .......................................................................................................... 12

ACKNOWLEDGMENTS ................................................................................................................ 13

ABBREVIATIONS ....................................................................................................................... 14

1 INTRODUCTION .................................................................................................................... 15

1.1 RESEARCH AIM, OBJECTIVES AND ASSUMPTIONS ......................................................... 15

1.2 RESEARCH CHALLENGES ............................................................................................... 16

1.3 INTENDED RESEARCH CONTRIBUTIONS .................................................................... 18

1.4 RESEARCH METHODOLOGY .......................................................................................... 19

1.5 THESIS STRUCTURE ....................................................................................................... 20

2 QUALITY REQUIREMENTS: DEFINITIONS, CLASSIFICATIONS AND TREATMENTS .......... 22

2.1 INTRODUCTION ............................................................................................................... 22

2.2 QUALITY REQUIREMENTS CLASSIFICATION MODELS .................................................. 23

2.2.1 ISO 25000 and ISO/IEC 9126 Standard Model .............................................................. 23

2.2.2 McCall’s Quality Model .............................................................................................. 26

2.2.3 Boehm’s Quality Model ............................................................................................. 29

2.2.4 Sommerville’s Quality Model .................................................................................... 31

2.2.5 Dromey’s Quality Model .......................................................................................... 33

2.2.6 FURPS/FURPS+ Model ............................................................................................. 34

2.2.7 Roman’s Quality Taxonomy ....................................................................................... 34

2.2.8 Quality Model for Business Process Outsourcing (BPO) ........................................... 35

2.2.9 SERVQUAL Model for Web Application .................................................................... 37

2.3 QUALITY REQUIREMENTS DEFINITIONS USED IN THIS THESIS ............................... 37

2.4 QR CHALLENGING FEATURES ...................................................................................... 39

2.4.1 Intangible and Functionally Dependent ...................................................................... 40

2.4.2 Subjective Requirements ........................................................................................... 40

2.4.3 Relative and Application-Specific Requirements ....................................................... 40

2.4.4 Interconnected and Interrelated Requirements ........................................................... 40
5.3.3 Quality Requirement Description Schema .............................................. 83
5.4 QRMF Representation Schema Structure .................................................. 84
  5.4.1 Row One: Strategic Level ................................................................. 86
  5.4.2 Row Two: Tactical Level ................................................................. 87
  5.4.3 Row Three: Operational Level .......................................................... 87
  5.4.4 Column One: Goal View ................................................................. 88
  5.4.5 Column Two: Agent View ............................................................... 88
  5.4.6 Column Three: Structure View ......................................................... 89
  5.4.7 Column Four: Function View ........................................................... 89
  5.4.8 Column Five: Architecture View ...................................................... 89
  5.4.9 Column Six: Behaviour View ............................................................ 90
5.5 The Framework Cells — Representing Different Perspectives of QR ............ 90
  5.5.1 Strategic Goal, Cell (1,1) ................................................................. 91
  5.5.2 Strategic Agent, Cell (1,2) ............................................................... 93
  5.5.3 Strategic Structure, Cell (1,3) ............................................................ 93
  5.5.4 Strategic Function, Cell (1,4) ............................................................ 93
  5.5.5 Strategic Architecture, Cell (1,5) ....................................................... 93
  5.5.6 Strategic Behaviour, Cell (1,6) .......................................................... 94
  5.5.7 Tactical Goal, Cell (2,1) ................................................................. 94
  5.5.8 Tactical Agent, Cell (2,2) ............................................................... 95
  5.5.9 Tactical Structure, Cell (2,3) ............................................................ 95
  5.5.10 Tactical Function, Cell (2,4) ........................................................... 95
  5.5.11 Tactical Architecture, Cell (2,5) ....................................................... 96
  5.5.12 Tactical Behaviour, Cell (2,6) ......................................................... 96
  5.5.13 Operational Goal, Cell (3,1) ........................................................... 97
  5.5.14 Operational Agent, Cell (3,2) .......................................................... 97
  5.5.15 Operational Structure, Cell (3,3) ...................................................... 97
  5.5.16 Operational Function, Cell (3,4) ....................................................... 98
  5.5.17 Operational Architecture, Cell (3,5) ............................................... 98
  5.5.18 Operational Behaviour, Cell (3,6) .................................................... 99
5.6 The QRMF Contributions and Features .................................................... 99
5.7 Summary .................................................................................................... 100

6 The QRMF Meta-Model ................................................................................. 102
  6.1 Introduction ............................................................................................... 102
  6.2 QRMF Meta-Model Structure ................................................................. 102
    6.2.1 The Meta-Meta Level ........................................................................ 103
6.2.2 The Meta-Level ................................................................. 104
6.2.3 The Domain Level ............................................................ 104
6.2.4 The Instance level ............................................................ 104

6.3 QRMF Meta-Model ............................................................... 104
6.3.1 Goal package ................................................................. 105
6.3.2 Agent package ............................................................... 105
6.3.3 Structure package ........................................................... 106
6.3.4 Function package ........................................................... 106
6.3.5 Architectural package ...................................................... 106
6.3.6 Behaviour package .......................................................... 107

6.4 The Goal Meta-Model .......................................................... 107
6.5 Agent Meta-Model ............................................................... 109
6.6 Structure Meta-Model .......................................................... 111
6.7 Function Meta-Model ........................................................... 112
6.8 Architecture Meta-Model ..................................................... 114
6.9 Behaviour Meta-Model ........................................................ 116
6.10 Summary ............................................................................. 118

7 THE QRMF MODELLING PROCESS ............................................. 120
7.1 Overall Modelling Process for the Framework ......................... 120
7.2 Phase One: Capture Strategic Requirements .......................... 122
7.3 Phase Two: Capture Tactical Requirements ........................... 125
7.4 Phase Three: Capture Operational Requirements .................... 132
7.5 Summary ............................................................................. 141

8 CASE STUDY: APPLYING THE QRMF TO A RADIOLOGY REPORTING SYSTEM .......... 143
8.1 Case Study Hypothesis and the Evaluation Measures ................ 143
8.2 Background for the Pilot Study ............................................. 144
8.3 Considerations for the Case Study ........................................ 148
8.4 Conducting the Case Study .................................................... 149
8.4.1 Phase One: Strategic Level .............................................. 149
8.4.2 Phase Two: Tactical Level ............................................... 154
8.4.3 Phase Three: Operational Level ....................................... 157
8.5 Case Study Analysis and Discussion ...................................... 166

9 APPLICABILITY AND COMPARISON WITH RELATED WORKS .......................... 168
9.1 Introduction ........................................................................... 168
9.2 Evaluation of the Applicability of the QRMF Meta-Model ......... 168
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.2.1</td>
<td>Applicability to Convey Generic Concepts for Goal Modelling View</td>
<td>169</td>
</tr>
<tr>
<td>9.2.2</td>
<td>Applicability to Convey Generic Concepts for architecture Modelling View</td>
<td>170</td>
</tr>
<tr>
<td>9.3</td>
<td>EVALUATION OF THE APPLICABILITY OF THE QRMF MODELLING PROCESS</td>
<td>171</td>
</tr>
<tr>
<td>9.3.1</td>
<td>Applicability of the Modelling Process to Adopting the QRMF Via Object-Oriented Approach:</td>
<td>172</td>
</tr>
<tr>
<td>9.3.2</td>
<td>Applicability of the Modelling Process to the Adoption of the QRMF by the Process-Oriented Approach:</td>
<td>175</td>
</tr>
<tr>
<td>9.3.3</td>
<td>Summary of the Analysis</td>
<td>177</td>
</tr>
<tr>
<td>9.4</td>
<td>EVALUATION OF QRMF AGAINST RELATED WORK</td>
<td>179</td>
</tr>
<tr>
<td>9.4.1</td>
<td>Process-Oriented Approaches to QR</td>
<td>179</td>
</tr>
<tr>
<td>9.4.2</td>
<td>Extension of Requirements Models to Address QR</td>
<td>182</td>
</tr>
<tr>
<td>9.5</td>
<td>SUMMARY</td>
<td>185</td>
</tr>
<tr>
<td>10</td>
<td>CONCLUSION</td>
<td>187</td>
</tr>
<tr>
<td>10.1</td>
<td>THESIS CONTRIBUTIONS</td>
<td>187</td>
</tr>
<tr>
<td>10.2</td>
<td>LIMITATIONS OF THE RESEARCH</td>
<td>189</td>
</tr>
<tr>
<td>10.3</td>
<td>FUTURE WORK</td>
<td>190</td>
</tr>
</tbody>
</table>

REFERENCES .................................................................................................................. 191

APPENDIX A.................................................................................................................. 201

APPENDIX B.................................................................................................................. 204
**List of Figures**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Thesis Structure and the Relationships Between the Chapters</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>ISO 9126 Quality Model [54]</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>McCall’s Quality Model [56]</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>McCall’s Quality Factors and Criteria of Software Quality [56]</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td>Boehm’s Quality Model [20]</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>Sommerville’s Quality Model [46]</td>
<td>32</td>
</tr>
<tr>
<td>7</td>
<td>Principles of Dromey’s Quality Model [48]</td>
<td>33</td>
</tr>
<tr>
<td>8</td>
<td>Technology-Driven Customer Quality Framework for BPO [50]</td>
<td>36</td>
</tr>
<tr>
<td>9</td>
<td>A Catalogue of Some NFR Types [19]</td>
<td>54</td>
</tr>
<tr>
<td>10</td>
<td>Method Catalogue of Confidentiality [19]</td>
<td>54</td>
</tr>
<tr>
<td>11</td>
<td>Softgoal Interdependency Graph [19]</td>
<td>55</td>
</tr>
<tr>
<td>12</td>
<td>Strategic Dependency Model Example [14]</td>
<td>57</td>
</tr>
<tr>
<td>13</td>
<td>A Strategic Rationale Model Example [14]</td>
<td>58</td>
</tr>
<tr>
<td>14</td>
<td>KAOS Notation and Concepts [67]</td>
<td>59</td>
</tr>
<tr>
<td>15</td>
<td>Requirements Modelling Framework and QR Modelling Issues</td>
<td>63</td>
</tr>
<tr>
<td>16</td>
<td>Zachman Framework: Executive, Owner, Designer Perspective [137]</td>
<td>67</td>
</tr>
<tr>
<td>17</td>
<td>Matrix for Selecting Complementary Modelling Languages [120]</td>
<td>70</td>
</tr>
<tr>
<td>18</td>
<td>Illustration of the QRMF Components</td>
<td>75</td>
</tr>
<tr>
<td>19</td>
<td>QR Description Schema</td>
<td>83</td>
</tr>
<tr>
<td>20</td>
<td>Generic Perspectives of the Framework Representational Schema</td>
<td>86</td>
</tr>
<tr>
<td>21</td>
<td>The QRMF Representational Schema</td>
<td>92</td>
</tr>
<tr>
<td>22</td>
<td>The Meta-Modelling Levels of QRMF</td>
<td>103</td>
</tr>
<tr>
<td>23</td>
<td>An Overview of the QRMF Meta-Models Representing Requirement Views</td>
<td>105</td>
</tr>
<tr>
<td>24</td>
<td>Goal Meta-Model</td>
<td>108</td>
</tr>
<tr>
<td>25</td>
<td>Relationship Type for Goal Meta-Model</td>
<td>108</td>
</tr>
<tr>
<td>26</td>
<td>Agent Meta-Model</td>
<td>110</td>
</tr>
<tr>
<td>27</td>
<td>Structure Meta-Model</td>
<td>112</td>
</tr>
<tr>
<td>28</td>
<td>Function Meta-Model</td>
<td>113</td>
</tr>
<tr>
<td>29</td>
<td>Architecture Meta-Model</td>
<td>115</td>
</tr>
<tr>
<td>30</td>
<td>Behaviour Meta-Meta-Model</td>
<td>117</td>
</tr>
<tr>
<td>31</td>
<td>QRMF Overall Modelling Process</td>
<td>121</td>
</tr>
<tr>
<td>32</td>
<td>A Blueprint of the Framework Modelling Process Phases and Steps</td>
<td>122</td>
</tr>
<tr>
<td>33</td>
<td>Steps for Capture of Strategic Requirements Phase</td>
<td>125</td>
</tr>
<tr>
<td>34</td>
<td>Step 1, Basic Relations to Obtain the Static Model of the Tactical Level</td>
<td>127</td>
</tr>
<tr>
<td>35</td>
<td>The Process of Constructing the Static Model of Tactical Level Using 1st Framework</td>
<td>129</td>
</tr>
<tr>
<td>36</td>
<td>An Example of a Static Model of Tactical Level Using SR of 1st Framework</td>
<td>130</td>
</tr>
</tbody>
</table>
List of Tables

TABLE 1 SIG CONTRIBUTION TYPE .......................................................... 55
TABLE 2 SOFTGOAL STATUS LABELS .......................................................... 56
TABLE 3 MAPPING REQUIREMENT ENGINEERING VIEWS INTO ZACHMAN FRAMEWORK .......................................................... 67
TABLE 4 MAPPING BUSINESS PROCESS SCHEMA ONTO ZACHMAN MODELS [120] .......................................................... 69
TABLE 5 MULTI-PERSPECTIVE MODELLING OBJECTIVES .......................................................... 71
TABLE 6 MAPPING ZACHMAN’ S ROWS TO SOFTWARE DEVELOPMENT PHASES .......................................................... 72
TABLE 7 MAPPING ZACHMAN’ S ROWS TO REQUIREMENT PROCESS .......................................................... 72
TABLE 8 QR CONCERNS RELATED TO A SOFTWARE PRODUCT .......................................................... 78
TABLE 9 QR CONCERNS RELATED TO THE DEVELOPMENT PROCESS .......................................................... 78
TABLE 10 QR DESCRIPTION FORM FOR EARLY REQUIREMENT .......................................................... 80
TABLE 11 QR DESCRIPTION FORM FOR LATE REQUIREMENT .......................................................... 82
TABLE 12 MODELLING CONCEPTS OF GOAL MODELLING VIEW .......................................................... 169
TABLE 13 MODELLING CONCEPTS OF ARCHITECTURE MODELLING VIEW .......................................................... 170
Abstract

In recent years, a considerable amount of research has been conducted in modelling Non-Functional Requirements (NFR) or Quality Requirements (QR). However, in comparison with functional requirements (FR) modelling, QR models are still immature and have not been widely adopted. The fundamental reason for this shortfall outlined in this thesis is that the existing QR modelling approaches have not adequately considered the challenging nature of QRs. In this thesis, this limitation is addressed through integrating QR modelling with FR modelling in a multi-perspective modelling framework. This framework, thus called QRMF (Quality Requirements Modelling Framework), is developed offering a process-oriented approach to modelling QR from different views and at different phases of requirement. These models are brought together in a descriptive representation schema, which represents a logical structure to guide the construction of requirement models comprehensively and with consistency. The research presented in the thesis introduces a generic meta-meta model for QRMF to aid understanding the abstract concepts and further guide the modelling process; it offers a reference blueprint to develop a modelling tool applicable to the framework. QRMF is supported by a modelling process, which guides requirement engineers to capture a set of complete, traceable and comprehensible QR models for software system. The thesis presents a case study, which demonstrates the practicality and applicability of the QRMF. Finally, the framework features were demonstrated through comparing and contrasting related approaches found in the literature.
Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.
Copyright Statement

1. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

2. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

3. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions can not and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

4. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=487), in any relevant Thesis restriction declarations deposited in the University Library, The University Library's regulations (see http://www.manchester.ac.uk/library/aboutus).
Acknowledgments

I would like to start by thanking ALLAH (swt) for getting me thru this huge undertaking and for all the open doors.

Would like to express my special appreciation and thanks to my advisor Professor Dr. Liping Zhao, you have been a tremendous mentor for me. Thank you for encouraging my research and thank you for supporting me during my personal ordeals. I would like to thanks Dr. Sam Supakkul and Dr. Pedro Sampaio for their valuable advice and comments.

Words cannot express how grateful I am to my Father for believing and supporting my dreams, this PhD is a tribute to your job as a father. Grateful thanks to my beloved mother for her continuous care, prayers and support. An honourable gratitude goes to my husband, my best friends and my mentor Chalmadi for understandings, motivating and supporting me in completing this research with a peace in my life. To my Zamzam. It took a while, but here you are. You fill my heart of joy and hope, to which I can only say AlhamduliALLAH. I have encountered many inspiring women in my life, and I wish one day to be the same for you. Mommy loves you.

Thanks to my siblings specially to Hassan for sharing the journey with me (and for being the coolest babysitter), to my sister and confidant Esraa. Thanks to my late Grandmother Aicha (ALLAH God rest her soul) for paving the road of my ambitions. Thanks to my Granma Zaynab, aunty Fatma, and aunty Abla, to my uncles specially uncle Ibraheem who always addressed me as Dr. Would also like to thank all of my friends who supported me and incented me to strive towards my goal. Thanks to Suha taking care of Zamzam during my last chapters.

My Friends Dr. Mona Zaki, Dr. Dalal Abasi, Hana Ameer, Dr. Safaa Jambi, Dr. Reem Aljendan, Khadijaa Bakur, Prof. Esma Bahurmoz, Eman Janbi, Asma and Harun Gencer for their support love and care during my times of doubt and stress. To Liz Mayne, thank you so very much for walking with me through the storms. Thanks to Dr. Abraham Nieva de la Hidalga, Dr. Habibullah Ameno, Dr. James Nash and my officemates for the time and effort to review and discuss my work, my goals and the crazy in between.

Finally, I would like to extend the thanks to the Saudi Arabian Cultural Bureau, Saudi Arabian Ministry of Higher Education and King Abdulaziz University for generous sponsorship and support, your excellent work represents the aim of our Nation.
Abbreviations

ANSI  American National Standards Institute
AORM  Agent-Oriented Requirements Modelling
ASQ  American Society for Quality
BPMN  Business Process Modelling Notation
BFORM  Business Process-oriented Requirements Modelling
CBSP  Component-Bus-System-Property
DFD  Data Flow Diagram
ER  Entity Relationship
FRs  Functional Requirements
GORM  Goal-Oriented Requirements Modelling
HIS  Hospital Information System
KAOS  Knowledge Acquisition in Automated Specification
LEL  Language Extended Lexicon
NFR  Non-functional Requirements
OCL  Object Constraint Language
OORM  Object-oriented Requirements Modelling
PACS  Picture Archiving and Communication System
PeRF  Performance Requirements Framework
PMR  Patient Medical Records
QADA  Quality-driven Architecture Design and quality Analysis
QASAR  Quality Attribute oriented Software Architecture
QRF  Quality Requirements of a software Family
QRID  Quality Requirements ID
QRMF  Quality Requirements Modelling Framework
QR  Quality Requirements
QUASAR  QUALity-driven System ARchitecture
RE  Requirements Engineering
RIS  Radiology Information System
SD  Strategic Dependency
SDL  Specification and Description Language
SIG  Softgoal Interdependency Graph
SQuaRE  Software Quality Requirements and Evaluation
SR  Strategic Rationale
SRS  Software Requirements Specification
UML  Unified Modelling Language
TTR  time-to-repair
EIS  Enterprise Information System
BPO  Business Process Outsourcing
BPM  Business Process Modelling
IHE  Integrating Healthcare Enterprise
VR  Voice Recognition
RRP  Radiology Reporting Process
RTT  Reporting Turnaround Time

Rules for abbreviations

- The term is expressed in full the first time it is used.
- If the term is used in more then one chapter, the term will be expressed in full the first time it used in each chapter.
- Tables and figures may introduce new abbreviations which need to be interpreted in context.
- Reference to other works may introduce additional abbreviations.
1 Introduction

Requirements engineering (RE) plays a critical role in the software engineering life cycle. Research has shown practical evidence that failures in software projects are largely due to poor requirements specification [1]. The RE community has classified software requirements into functional and non-functional requirements (FRs and NFRs). FRs describe what the software will do, whereas NFRs describe how well the software will perform [2]. Software systems are built mainly to solve real-world problems which are more non-functionally oriented, such as poor productivity, slow processing and unhappy customers [3]. Moreover, the NFRs constrain the realization of FRs as well as playing an essential role in design decisions [4]. Therefore, the specification of software functionality can not be useful or complete unless specification of the necessary NFRs is considered [3, 5]. The NFRs are referred to as the Quality Requirements (QR) of software systems [3, 4, 6-8] and are mainly associated with user satisfaction [9-11] and business goals [4, 7, 8], which are the most significant indicators of software success. Addressing QR at an early stage of software development is vital for successful development as it saves time, cost, resources and the effort of repairing software defects [3, 9).

Modelling raises the abstraction level to express the core essentials. This aids understanding the current reality of the existing system and domain features, and also fosters creativity in designing the new system [12]. Modelling is the most significant medium to elicit, define, analyze, validate, verify and communicate requirements. This is true because at the early stage of the software life cycle stakeholders and requirement engineers do not have a clear idea about real requirements. The graphical representation of modelling requirements helps to express, discuss and understand requirements among stakeholders with different backgrounds. Furthermore, it is cheaper and faster to incorporate changes in requirement models than later during implementation.

The significance of modelling QR at an early stage in the software life cycle has been recognized in the RE literature [13] [14]. This thesis presents a new approach to QR modelling at the RE stage of the software life cycle.

1.1 Research Aim, Objectives and Assumptions

The aim of this thesis is to address the limitations of current QR modelling approaches to support the construction of comprehensive QR models based on Zachman framework; the framework is generic can be adopted in a wide range of requirement
modelling techniques. In order to achieve this aim the research seeks to answer the following research questions:

**Question 1:** What are the limitations of current QR modelling approaches?

**Question 2:** What are the components of a framework that can overcome these limitations?

**Question 3:** How can the adequacy of the framework be demonstrated?

One of the fundamental assumptions of this research is that the RE literature is rich in powerful and widely adopted modelling languages, and there is no need for a new modelling notation to address QR. However, what is missing is a framework that guides incorporating QR concepts within FR models using current requirement modelling approaches. The benefit of this is that our proposal can be adopted using the modelling approach and languages familiar to requirements analysts, which will increase the potential audience who may interested in adopting, validating or improving our proposal.

This research will pursue the following objectives in order to address the research questions and achieve the stated aim:

**Objective 1:** To design a multi-perspectives modelling framework that integrate QRs modelling and FR modelling.

**Objective 2:** To support this framework with a guided modelling process.

**Objective 3:** To demonstrate the viability and adoptability of the framework through a case study.

**Objective 4:** To demonstrate the framework features against similar work

### 1.2 Research Challenges

QR are major drivers in software development as they determine the design complexity of software systems. The importance of QR for software systems has been recognized since the seventies [15]. Since then a considerable amount of research has been conducted into modelling QR. However, in comparison with FRs, QR modelling are still immature and has not been widely adopted [3, 16]. Based on a comprehensive literature review (introduced in Chapters 2, 3 and 4), this research argues that none of the existing QR modelling approaches can comprehensively and completely document and communicate QR for two reasons. The first reason is related to the challenging nature of QR which described as Intangible and functionally dependent, subjective, application-specific, system Level, interconnected and interrelated [7] [17] [18] [6]. It is difficult to
address all QR features without a structured and guiding framework that integrates the QR model with the functional requirements model. The second reason is that all the existing QR modelling approaches focus on a single or limited set of requirement views. Therefore, this research bridges this limitation by offering a multi-perspective modelling framework for QR and supporting this framework with a guided modelling process.

**Challenge 1: The Nature of Quality Requirements**

QR are described as intangible, functionally dependent, subjective, relative, interrelated, complex and cross-disciplinary requirements [17, 19]. More about these features can be found in Chapter 2. These features make expressing QR in a complete and comprehensive fashion difficult. Their intangibility and functional dependence makes QR prone to be forgotten in the midst of the requirement development process. A graphical modelling approach that annotates QR can resolve this issue. The subjective and relative features of QR refers to the different concerns and meanings of a QR. In another words, a QR means different things to different stakeholders and from different requirement views. A precise definition of QR can solve this issue. As to the complex, interrelated and cross-disciplinary features, it is not enough to express a QR comprehensively in a single view of a requirement or at one phase of the requirements process. Incorporating expressed QR into all requirement modelling views at early and late phases can solve this issue. Tracing the evolution of QR among different requirement models is important for requirement validation and completeness checking. This can be addressed by tracing the interrelated QR with a sequence of IDs or a tree-structured model.

**Challenge 2: Current QR Modelling Approaches**

The QR literature proposes a wide range of quality classification schemes [20] [21] [22], in a product-oriented approach to QR. More about the QR classification models can be found in Chapter 2. Although those models provide a reference point or catalogue to define QR [23], they are generic and difficult to incorporate or customize into a specific context or software setting without detailed guidance [17, 19].

These limitations are addressed to some extent through the process-oriented approach to QR, which provides a detailed process to capture the QR of a software system. Although some of these approaches have been widely adopted, such as the NFR and i* frameworks, QR are addressed only in specific views of requirements. For example, the NFR framework [19] focuses on a goal view, i* [14] focuses on a goal and agent view; and
both offer a static model of requirements. It is insufficient to fit requirement specifications in a single requirement model [24-26] as a single model can not capture enough concepts of all requirement views. This also applies to QR, which are more complex and challenging to model than functional requirements.

Another research direction of the process-oriented approach is extending the commonly known modelling frameworks and notations [9, 27-29] with QR concepts. This direction focuses on a single requirement perspective and is limited to a specific modelling language.

1.3 Intended Research Contributions

This research is intended to deliver the following contributions to QR modelling.

**Contribution 1:** To identify the limitations of current QR modelling, through investigating the QR natural characteristics of QR that challenge QR modelling, and then discover to what extent the current QR modelling frameworks address the identified challenges; based on this, the framework's key features are identified.

**Contribution 2:** To introduce a novel QR modelling framework that facilitates constructing a complete and comprehensive QR model. The QRMF will offer the following components:

- A multi-perspective modelling framework based on a generic two-dimensional schema. The columns represent the universal interrogatives to a phenomenon, which in this research reflects the entire requirement modelling views including goal, agent, structure, function, architecture and behaviour. The rows represent the transformation from early to late requirement phases including strategic, tactical and technical phases.
- The QRMF meta-model illustrates the core concepts and relationships of the multi-perspectives QR models regardless of the modelling notation that is used.
- The QRMF modelling process offers a constructive framework for building QR models. The modelling process offers generic guidelines on the use of the framework without restricting the requirement modelling approach.

This study uses a radiology reporting system in the healthcare domain as an example of how the QRMF can be applied to a specific project, and to demonstrate the
utility and adaptability of the framework using different requirements modelling approaches and notations. QR

**Contribution 3:** Visual annotation of QR concepts in requirement models. This is significant to avoid overlooking intangible QR and to facilitate communicating QR in a comprehensible fashion. The framework uses annotations of QR concepts in existing requirement model languages that are familiar to requirements analysts, i.e. it extends existing modelling languages with QR concepts.

**Contribution 4:** Traceability of QR through a sequence of QRIDs. A unique QRID is generated to each QR elicited at an early stage of requirements, then a hierarchical sequence number is attached to the QR generated through goal refinement, functional composition, objects generalization or following the flow of process.

### 1.4 Research Methodology

A constructive research approach has been adopted to advance the practical and theoretical evidence in developing the framework. The objective of constructive research is to produce a novel solution to practical and theoretical problems through the construction of models, diagrams, constructs, process, methods, framework, etc. [30-32]. The constructive research approach is widely used in software engineering and computer science [31, 32]. The constructive research method involves four major activities: problem analysis and background study; design of the framework; empirical investigation; and evaluation.

1. **Problem analysis and background study:** This stage aims to identify the problem to be solved and obtain a good understanding of the problem. Several areas related to the research objectives were reviewed with regard to their possible contribution to QR modelling in RE. This investigation is grouped into three main areas: QR definitions and classification models in Chapter 2, requirement modelling frameworks in Chapter 3, and multi-perspectives requirement modelling frameworks in Chapter 4.

2. **Solution design:** This stage involves innovation to design a solution to the identified problem, i.e. constructing a solution idea, design of a framework, development of tools, etc. This may be a heuristic process, and also includes theoretical justification. This research offers a novel solution to model QR which consists of three original components: a multi-perspective framework for QR that
offers a descriptive representation schema of the entire RE modelling views, as
detailed in Chapter 5; meta-models express the framework modelling concepts at
an abstract level, as described in Chapter 6; and a framework modelling process
that guides the construction of the models, as explained in Chapter 7.

3. **Empirical investigation:** This stage aims to demonstrate that the solution to the
identified problem works; it also tests the features of the proposed solution
through a case study. Ideally constructive research requires a case study in an
industrial setting. However, the empirical demonstration of this research is
conducted through a case study in a research context. The background of the case
study is developed both through thorough investigation of practical systems and
the literature. The case study is conducted following the guidelines for software
engineering tools and methods [33] [34]. The case study is detailed in Chapter 8.

4. **Evaluation:** This stage aims to validate features of the proposed solution against
related work in the RE literature to demonstrate the strength and weakness of the
solution. It also examines the scope of applicability of the proposed solution
adopting different RE approaches and modelling notations. These two types of
demonstration are detailed in Chapter 9.

1.5 **Thesis Structure**

The thesis consists of ten chapters and two appendices. The overall structure of
the thesis is illustrated in Figure 1. The first chapter represents the conceptual framework
and methodology that underpin this research project to meet the identified objectives. The
remaining chapters represent the research undertaken following the methodology
described earlier in this chapter. The remainder of this thesis is structured as follows:

![Figure 1 Thesis Structure and the Relationships Between the Chapters](image-url)
**Chapters 2, 3 and 4:** complement the conceptual framework of the research project by introducing the fundamental research areas that underpin the design of the QRMF. These chapters review the area of knowledge and describe how they are integrated for the development of the QRMF.

**Chapter 5:** introduces the details of QRMF and the rationale behind the framework design. The chapter presents the details of a complete description of QR then integrates that description into the representational schema.

**Chapter 6:** presents a higher-level of abstraction of the framework's meta-concepts. Four abstraction levels are illustrated in this chapter, and a detailed description of the meta-meta-model is provided.

**Chapter 7:** presents a constructive method for elaborating a robust and consistent model of QR integrated with requirements models. The process offers an incremental process to identify, analyze and validate requirements with the related QR in an integrated multi-perspective modelling view.

**Chapter 8:** demonstrates the ability of the QRMF through the case study of a radiology reporting system.

**Chapter 9:** demonstrates the applicability of the framework meta-meta-model and modelling process using different requirement modelling approaches and notations. It also demonstrates the framework's features and limitations against related work.

**Chapter 10:** summarizes the research contributions and benefits of the framework and reviews them against the stated research objectives. It also offers suggestions for further work.

**Appendix A:** complements Chapter 2 by listing 227 different qualities found in the literature.

**Appendix B:** complements Chapter 8 by presenting tables used for analyzing the constructed models using different RE approaches.
2 Quality Requirements: Definitions, Classifications and Treatments

2.1 Introduction

Crosby defines quality as “conformance to requirements” [35]. The definitions of quality as the “totality of features and characteristics of a product or service that impact its ability to satisfy given needs” was standardized in 1978 by the American National Standards Institute (ANSI) and the American Society for Quality (ASQ). From the 1980s, researchers agreed that quality is important in satisfying customer expectations [19, 36, 37] and business goals [8, 38, 39], requirements that are significant for successful software development. Quality requirements (QR) are referred to as NFRs [8, 38, 40] that define how well a software system accomplishes the requirements [41] [42]. Software systems are built to solve real-world problems which tend to be non-functional oriented, such as poor productivity, slow processing and unhappy customers [40]. Time, reliability, efficiency, accessibility, responsiveness, flexibility, ease of use and usability are other examples of QR. The QR constrain the realization of software functionality and are an essential factor of design decisions [38]. Therefore, the specification of software functionality can not be useful or complete unless specification of the necessary QR is considered [40] [43]. In addition, addressing QR at an early stage of software development is vital for successful development as it saves time, costs, resources and effort in fixing software defects [40, 44].

The literature on software quality offers a range of quality classification models that define and divide QR into several types. These models represent the product-oriented approach to quality requirements. Examples of common classification schemes and models are summarized in section 2.2. Section 2.3 provides definitions of the quality characteristics used in this thesis. These definitions are highlighted to aid understanding the framework; however, as QR can not be defined out of context, these definitions may vary depending on the application type, domain and scenario. Section 2.4 outlines some of QR’ natural features that challenge their treatment in software systems. Section 2.5 highlights the benefits and limitations of quality classification models. Finally, section 2.5 provides a summary of this chapter.
2.2 Quality Requirements Classification Models

QR classification, or the product-oriented approach, focuses on defining, measuring and evaluating the QR of a developed software product [23]. This approach has received exclusive attention in the literature where a wide range of QR approaches has been developed, standardized and adopted. This approach classifies QR according to different levels of characteristics and sub-characteristics with identified relationships among them, and a matrix for QR measurement and evaluation. Most of these classification models and standards offer hierarchical catalogues as guides to the treatment of QR. The product-oriented approaches to QR are quantitative in the sense that they study quantitative metrics for measuring the degree to which a software system satisfies NFRs [13]. Software Quality Requirements and Evaluation (SquaRE) of ISO/IEC 9126 and 14598 standards [45], McCall’s quality model [22], Boehm’s quality tree model [20], Sommerville’s non-functional classification [46], FURPS [47], Dromey Model[48], Roman’s NFR taxonomy [49], quality framework for BPO [50] and SERVQUAL [51] are examples of the most popular classification schemes. This section provides an overview of these schemes.

2.2.1 ISO 25000 and ISO/IEC 9126 Standard Model

The Software Quality Requirements and Evaluation (SQuaRE) described in ISO 25000 standard [52] has five divisions: quality management, quality model, quality measurement, quality requirements and quality evaluation. This standard takes a system perspective and suggests specifying requirements as measures and associated target values. The quality management division defines all common models, terms and definitions in SQuaRE standards. This standard also provides guidance for managing software product requirements specifications. The quality model division includes the definitions and measures of QR characteristics and sub-characteristics found in the ISO/IEC 9126 classification model. The quality measurement division includes a reference model and mathematical definitions that guide specifying the quality measures of a software product. The quality requirements division help to identify and specify QR. Finally, the quality evaluation provides guidelines for software product evaluation. This section briefly reviews the external quality model described as ISO/IEC 9126.

The standard quality model in ISO/IEC 9126-1 [21] categorizes QR in a hierarchy of six quality characteristics and and 26 sub-characteristics, which are further decomposed into attributes. The classification provides a basis for quantifying the quality requirements in terms of software quality measures. The measurement attributes appear
at the bottom of the hierarchy and values are computed using metrics [53]. The following is a brief outline of these characteristics. Figure 2 shows the taxonomy of quality characteristics in the ISO 9126-1 standard.

**Functionality** refers to the ability to satisfy the stated needs. It consists of five sub-characteristics: suitability, accuracy, interoperability, security and functionality compliance. Suitability refers to the ability of the software to provide a set of functions appropriate for specified tasks. Accuracy refers to its ability to provide the agreed results. Interoperability is the ability to interact with specified systems. Compliance refers to the ability to adhere to specified standards, conventions or regulations. Security is the ability to prevent unauthorized access to the software’s functions and data.

**Reliability** refers to the ability to maintain performance level under stated conditions for a stated period of time. Its four sub-characteristics are maturity, fault tolerance, recoverability and reliability compliance. Maturity refers to the ability of the software to maintain limited frequency of failure. Fault tolerance is the ability to maintain a specified level of performance in the case of software faults or of infringement of its specified interface. Recoverability is the ability to re-establish its level of performance and recover the data directly affected by a failure, on time and the effort needed for this.
reliability compliance refers to the ability of the software to comply with standards, conventions or regulations in relation to reliability.

**Usability** is the ability to understand, learn and use the software interface. Its five sub-characteristics are understandability, learnability, operability, attractiveness and usability compliance. The first refers to the effort required by the user to recognize the software’s logical concept and its applicability. Learnability and operability refer to the effort of users to learn and to operate and control the software. Attractiveness refers to how good the software interface looks. Finally, usability compliance is the ability of the software to comply with standards, conventions or regulations in relation to interface.

**Efficiency** refers to the ability to meet the promised performance levels under specific resources and conditions, and it comprises time behaviour, resource utilization and efficiency compliance. Time behaviour refers to the ability of the software functions to maintain the agreed response times and throughput rates. Resource behaviour is the ability to maintain agreed performance rates and resources used in the software functions. Efficiency compliance is the ability of the software functions to comply with standards, conventions, or regulations in relation to efficiency.

**Maintainability** refers to the ability to modify and adapt to changes in the environment. Its five sub-characteristics are analyzability, changeability, stability, testability and maintainability compliance. The first refers to the effort needed to diagnose causes of software failure and modify them. Changeability is the effort needed for modification, fault removal or for environmental change. Stability refers to the ability to maintain the software functionality in the case of unexpected modifications. Testability is the effort needed for validating the modified software, and maintainability compliance is the ability to comply with standards, conventions, or regulations in relation to maintainability.

**Portability** refers to the ability of the software product to be transferred from one environment to another, comprising adaptability, installability, co-existence, replaceability and portability compliance. Adaptability refers to easy operation of the software in specified environments. Installability refers to the effort needed to install the software in a specified environment. Co-existence is the ability of the software to interact directly with other software. Replaceability refers to the effort needed to replace the software with other software in the same environment. Portability conformance refers to the ability to comply with standards, conventions or regulations in relation to portability.
The above standard ISO/IEC 9126 [55] also provides another level of classification that distinguishes three different views of quality: external quality, internal quality and quality in use. ISO 9126-2 defines external metrics for measuring and evaluating software execution behaviour that users can experience. ISO 9126-3 defines internal metrics for measuring and evaluating non-executable properties of software implementation, i.e. the QR recognized by software developers in order to create a good software product such as the quality of the design and code elements. ISO 9126-4 defines quality-in-use metrics for measuring and evaluating the effects on actual end users while the product is used in a specific context. Quality-in-use metrics are based on attributes important to users: effectiveness, productivity, safety and satisfaction [53]. This classification facilitates QR identification from different viewpoints and different levels: QR documentation and evaluation; QR operationalization and trade-off among them, identification of dependencies and conflict resolution; and finally QR management.

2.2.2 McCall’s Quality Model

McCall’s Quality Model/General Electrics Model [56] was developed for the US Air Force. It bridges the gap between users’ views and developers’ properties through focusing on the qualities of system developers and system development process. It consists of three major perspectives for defining and identifying the QR of a software product: product revision, product transition and product operations. The McCall quality model is shown in Figure 3.

**Product revision** refers to the quality factors that influence the ability to change the software product, including maintainability, flexibility and testability. Maintainability is the effort required to locate and fix a fault in the program within its operating environment; flexibility is the ability to make changes in the operating environment, and testability is the ability to validate the software requirements to ensure that the software is error-free and meets its specification.

**Product transition** refers to quality factors that influence the ability to adapt to new environments: portability, reusability and interoperability. Portability is the effort required to transfer the software from one environment to another; reusability is the ability to reuse the software in different contexts; and interoperability is the effort required for system components to work together.

**Product operation** refers to the quality factors that influence the software product operation in order to meet its specifications, namely correctness, reliability,
efficiency, integrity and usability. Correctness refers to the extent to which a program fulfils its specification, reliability to the system's ability not to fail, efficiency to optimum use of resources (CPU, storage or man-power), integrity to the protection of the program from unauthorized access, and usability to the ease of use of the software.

Figure 3 McCall's Quality Model [56]

The model details the three major quality perspectives in a hierarchy of 11 factors describing the external view of the software as seen by the users, 23 criteria describing the internal view as seen by the developer, and metrics for measurement. The metrics are relevant to quality criteria, quality factor, or the software product. This hierarchical classification offers a bottom-up quality assessment of a given software product. Figure 4 illustrates the hierarchical classification of these factors and criteria; it is followed by a brief definition of each criterion as specified in [56].

- **Traceability**: is the ability to trace the development of software requirements with respect to the development specification and operational environment.
- **Completeness**: is the ability to deliver all the requirements through the software's functions.
- **Consistency**: is the ability to offer uniform design, implementation techniques and notation of the software.
- **Accuracy**: is the ability of software to provide precise outputs.
- **Error tolerance**: is the ability to provide continuous operation under unexpected conditions.
- **Simplicity**: is the ability to easily understand the software implementation functions.
- **Modularity**: refers to the software structure, consisting of independent modules.
- **Generality**: is the ability to provide breadth to the performed functions.
- **Expandability**: is the ability of software to expand its data storage and computational functions.
- **Instrumentation**: is the ability to provide measurement of software usage and identification of errors.
- **Self-descriptiveness**: refers to the provision of explanations of the implementation of its functions.
- **Execution efficiency**: is the provision of minimum execution time.
- **Storage efficiency**: is the consumption of minimum storage during operation.
- **Access control**: is software that offers controlled access to its functions and data.
- **Access audit**: is software that offers an audit trail of its functions and data.
- **Operability**: determines procedures concerned with the software operation.
- **Training**: provides transition from current operation or initial familiarization.
- **Communicativeness**: means useful input and output that can be assimilated.
- **Software system independence**: determines dependence on the software environment (operating systems, utilities, input/output routines, etc.)
- **Machine independence**: determines dependence on the hardware system.
- **Communication commonality**: is the use of standard protocols and interface routines.
- **Data commonality**: is the use of standard data representation.
- **Conciseness**: is the implementation of a function with a minimum amount of code.

### 2.2.3 Boehm’s Quality Model

Boehm’s quality model [20] attempts to qualitatively define software quality through a set of attributes and metrics. Similar to the McCall quality model, Boehm’s model has a hierarchical tree structure with high-level, intermediate and primitive characteristics. The tree is shown in Figure 5. The highest level represents basic requirements related to the general utility of the software, including as-is utility, maintainability and portability. As-is utility refers to the extent to which the software can be used. Maintainability is the ease of identifying what needs to be changed, and how easy it is to perform and the change and retest. Portability refers to the software’s ability to adjust easily to a new software environment. The as-is utility, maintainability, and portability are necessary (but not sufficient) conditions for general utility. The intermediate level represents seven expected quality factors (see Figure 5).

- **Portability**: this is a sub-characteristic of general utility characteristics. It refers to the ability of the software to work in different environments, i.e. database and operating systems.
- **Reliability**: a sub-characteristic of as-is utility characteristics. It refers to the ability of the software to perform the expected functions at a satisfactory level and without defects.
- **Efficiency**: a sub-characteristic of as-is utility characteristics. It refers to the ability of the software to perform the specified functions with optimum use of processing and storage resources in the optimum time.
- **Usability/Human Engineering**: a sub-characteristic of as-is utility characteristics, referring to the software’s ease of use.
- **Testability**: a sub-characteristic of maintainability, referring to the ease of specifying software verification criteria to evaluate performance in relation to requirements.
Understandability: a sub-characteristic of maintainability characteristics, referring to the ability to easily comprehend the software with regard to purpose and structure.

Modifiability/Flexibility: a sub-characteristic of maintainability, referring to the ability of software to incorporate changes and revised requirements.

The lowest level represents the primitive characteristics of one or more metrics of software quality. At this level Boehm’s quality model includes the following fifteen characteristics:

- **Device independence**: the ability to execute the software on different hardware configurations.
- **Self-containedness**: the ability to perform all explicit and implicit functions within the software itself.
- **Accuracy**: the provision of sufficiently precise output to meet the specified requirements. This characteristic is necessary for reliability.
- **Completeness**: the ability to fully develop the agreed software component as presented in the design.
• **Integrity**: the ability to perform the desired functionality, and to produce reliable output despite some violation of the input variables and environment conditions specification.

• **Consistency**: to provide uniform notation, terminology, and symbology of software implementation including code structure, variable names, code comments and module settings.

• **Accountability**: the ability to measure the usage of software, time consumed to perform specific functions or resources used by specific functions.

• **Efficiency**: the ability to perform the agreed functionality with optimum use of CPU processing, storage resources and time.

• **Accessibility**: the ability to facilitate selective use of software parts, such as variable dimensioned arrays. Accessibility is necessary for efficiency, testability and human engineering.

• **Communicativeness**: the ability to understand and use the software. It is necessary for testability and human engineering.

• **Self-descriptiveness**: the ability to provide enough detail for the software objectives, assumptions, constraints, inputs, outputs, components, and revision status. This characteristic is necessary for both testability and understandability.

• **Structure**: to provide a definite pattern of organization of the software's interdependent parts.

• **Conciseness**: the ability to provide concise software implementation code, comments and instructions without excessive information.

• **Augmentability**: the ability to easily accommodate expansion in component computational functions or data storage requirements. It is a necessary characteristic for modifiability.

• **Legibility**: the ability of software functions to be easily discerned. It is necessary for understandability.

### 2.2.4 Sommerville’s Quality Model

Sommerville [46] offers a non-functional classification of features that are often raised through user needs, budget constraints, organizational policies, the need for interoperability with other software or hardware systems, or external factors. Sommerville’s quality model, shown in Figure 6, classifies QR based on sources including software product requirements, organizational requirements and external sources.
• **Product requirements**: specify the behaviour of the software, including usability requirements or how easy it is to use the system; this depends on the technical system components, its operators, and its operating environment; efficiency requirements, which refer to optimum use of system resources (memory and processor cycles) and include performance requirements (responsiveness and processing time) and space requirements (memory utilization); dependability requirements or the ability of the software not to cause physical or economic damage in the event of system failure; reliability requirements that specify the acceptable failure rate; and security requirements to prevent malicious access or damage to the system.

![Figure 6 Sommerville's Quality Model [46]](image)

• **Organizational requirements**: these are derived from organizational policies and procedures, and include operational process requirements, or how the system will be used; development process requirements referring to programming languages, development environment and process standards to be used for the software development; and environmental requirements referring to the operating environment.

• **External requirements**: these are external to the system and its development process and include regulatory requirements, such as a central bank setting the specifications for system approval; legislative requirements that ensure the system operates within the law; and ethical requirements that ensure the system can be accepted by users and the general public.
2.2.5 Dromey's Quality Model

Dromey proposed a quality evaluation framework that analyzes the quality of software through measuring the intangible quality properties [48, 57]. Each component produced in the software life cycle is associated with quality attributes, i.e. variables, functions and statements are considered as components of the software implementation model; and modules are considered as components of the design model. The software product components are classified into only four product properties, as simply identifying large numbers of properties does not provide a good basis for constructing a quality model:

- **Correctness property**: evaluates significant properties which, if violated, mean that the product will not meet the specification.
- **Internal property**: measures how well a component has been deployed according to its intended use or implementation requirements or how well it has been composed.
- **Contextual property**: deals with the external influences by and on the use of a component. It may involve contextual properties but with less violation of high-level quality attributes like maintainability.
- **Descriptive property**: measures the ability to understand and use the software requirements, designs, implementations and user interfaces.

This model focuses on the relationship between the quality attributes and the sub-attributes, as well as connecting software product properties with software quality attributes. The hierarchy is illustrated in.

![Figure 7 Principles of Dromey's Quality Model [48]](image-url)
2.2.6 FURPS/FURPS+ Model

The FURPS model [47] was developed by Robert Grady and Hewlett-Packard (HP) in 1987 to represent a classification model for software functional and non-functional attributes. FURPS is an acronym from its characteristics: Functionality, Usability, Reliability, Performance, Supportability. The following is a summary of these characteristics and their sub-characteristics:

- **Functionality**: defines what customers want, i.e. system-wide functional requirements and functions significant to software architecture. They include capability (size and generality of feature set), reusability (compatibility, interoperability, portability) and security (safety and exploitability).
- **Usability**: is the quality characteristic related to user interface issues including accessibility, aesthetics, consistency, documentation and responsiveness.
- **Reliability**: includes aspects related to availability, robustness, recoverability, survivability, predictability and accuracy (frequency and severity of error).
- **Performance**: includes aspects related to speed, efficiency, resource consumption (power, storage, RAM, cache, etc), throughput, capacity, recovery time and start-up time.
- **Supportability**: includes aspects to maintain and update the software including serviceability (maintainability, sustainability, repair speed), testability, flexibility (modifiability, configurability, adaptability, extensibility, modularity), installability, and localizability.

FURPS+ extends the FURPS model with additional concerns, including:

- **Design constraints**: specify or constrain the options for designing a system (such as I/O devices, databases).
- **Implementation requirements**: specify or constrain the coding or construction of a system (required standards, implementation languages, and resource limits).
- **Interface requirements**: specify an external item with which a system must interact, or constraints on formats or other factors used within such an interaction.
- **Physical requirements**: specify a physical constraint imposed on the hardware used to house the system (shape, weight, size, power).

2.2.7 Roman’s Quality Taxonomy

Roman [49] grouped the great diversity of types of non-functional requirement in the following taxonomy of QR concerns:
- **Interface concerns:** define the interaction between software components and the environment. These details should not affect what the program does, but the way components interact by capturing the syntax of procedure invocation, component communication, interrupt address and screen format.

- **Performance concerns:** cover a wide range of software runtime issues categorized into time/space efficiency, reliability, security and survivability. The time/space requirement includes response time, workload, throughput, and available storage space and productivity. The reliability requirement is related to the availability of physical components and the integrity of the information. The security requirement protects untarnished access to physical and logical system components. The survivability requirement related to the recovery of system functions and data in the case of unexpected conditions.

- **Operation concerns:** cover the QR in the context of operation and environment conditions including personal availability and skill level, accessibility for maintenance, physical constraints (size, weight, power), component distribution and environmental conditions.

- **Life cycle concerns:** cover two broad categories, QR relating to design and QR relating to software development. Failure to satisfy any of these requirements may not compromise the initial software outputs, but may increase life cycle costs and shorten the life span of the delivered software. The first category includes maintainability, enhanceability, portability, flexibility, reusability, production life span, upward compatibility, and integration into a family of products. The second category includes development time, resource availability and methodological standards.

- **Economic concerns:** cover considerations of immediate cost (development and setup) and long term costs (licences, training and maintenance).

- **Political concerns:** cover polices, standard and legal issues needed to be considered.

### 2.2.8 Quality Model for Business Process Outsourcing (BPO)

This model is a quality framework for BPO [50] which includes the major quality measurement criteria considered by consumers and providers. It has been widely adopted by well known software vendors such as ASP and HP. The framework comprises seven quality characteristics that enable the standardization, integration, automation and
innovation of technology that support BPO of software systems. Figure 8 illustrates the hierarchical model, and the quality characteristics are briefly defined as follows:

- **Reliability**: the ability to perform the promised service dependably and accurately. This characteristic can be measured through five sub-measures: on-time delivery, accuracy of results, accessibility, correct historical records and disaster recovery.

- **Tangibility**: is related to the QR of physical facilities, equipment, and application and appearance of personnel. These characteristics can be measured through six sub-measures: advanced technology, global expertise, application's user-friendly interface, ease of data reporting and extracting, application scalability and application interoperability.

- **Conformance**: is the degree to which a service's design and operating characteristics meet established standards. This characteristic can be measured through four sub-measures: systematic process design, consistent process delivery and management, efficiency and effectiveness, and added value.

- **Responsiveness**: the timeliness of service. Characteristics can be measured through four sub-measures: speed, competence, correction of errors and customer relationship.

- **Flexibility**: the ability to deal with changes, measured through four sub-measures: rescalability, availability of upgrades, innovation and transition.

- **Security**: the freedom from danger, risk or doubt, measured through confidentiality, physical safety and financial safety.
2.2.9 SERVQUAL Model for Web Application

SERVQUAL [51] focuses on the QR of web services and web applications, the items that have been identified as most important in relation to the quality of websites. The SERVQUAL model is also used to compare the perceptions of providers and customers in both B2C and B2B transactions. It has five characteristics with 22 sub-characteristics:

- **Tangibles**: the appearance of the website, navigation, search options, and structure.
- **Reliability**: the ability to judge the trustworthiness of the offered service and the organization performing the service.
- **Responsiveness**: the willingness to help customers and provide prompt service.
- **Assurance**: the ability of the website to convey trust and confidence in the organization behind it with respect to security and privacy.
- **Empathy**: the provision of caring, individualized attention to customers, including user recognition and customization.

2.3 Quality Requirements Definitions Used in this Thesis

QR cover a wide range of characteristics. 122 distinct QR were found in the above-mentioned models, and 227 QR definitions can be found in the literature. There is no agreed definition or complete list of QR, and no single universal classification model that accommodates all the QR needed for different types of application [19] and software scenarios. This research has chosen the following QR definitions and their associated measures. These definitions and measures will be used for reference to illustrate the framework features and its validation, as proposed in this thesis.

**Reliability**

**Definition**

The ability to perform promised functions within a specified time [50, 58] and under agreed conditions; In the other words it assures delivery of the promised requirements [59] and a non-stopping system [60]. Reliability rate is measured through the number of successful invocations, as shown by the following formula: [61]

\[
R = 1 - \frac{P(\text{success})}{N}
\]

Where \( N \) is the total number of invocations

\( P(\text{success}) = \frac{\text{number of successful execution}}{N} \)

The target is to achieve higher reliability rate
### Time

**Definition**
The time needed for a specific service or execution. It can be measured by response time, throughput and latency [58]. Response time is the maximum time required to complete a function [58]. Throughput is the number of requests served in a specific period of time. Latency represents the actual time between the sending of a request and receiving a response [58]. Good response time implies higher throughput and lower latency values. Time can be measured through the following attributes [61]:

**Measure [61]:**
- **Response time**: a specified time unit
- **Throughput**: the number of successful service in units of a specified period of time
- **Latency**:
  \[ \text{Latency} = \text{to}/p(X) - \text{ti}/p(X) \]
  where \( \text{ti}/p(X) \) is the timestamp when the service \( X \) is invoked
  \( \text{to}/p(X) \) is timestamp when the service \( X \) is delivered.

The target is to achieve a service or execution in minimum time.

### Availability

**Definition**
The ability to invoke a function at any point in time. It is also associated with time-to-repair (TTR), which is the time taken to repair a service. Availability is related to accessibility and reliability [58, 63]. Availability can be measured by one of the following formulae:

**Measure [58, 63]:**
- **Availability**: \( \frac{C(X)}{N} \)
  Where \( C(X) \) is the number of successful executions
  \( N \) is the total number of invocations.

\[ \text{TTR} = \text{t-restart}(X) - \text{t-failed}(X) \]
Where \( \text{t-failed} \) is the timestamp when the service \( X \) failed
\( \text{t-restart} \) is timestamp when service was restarted.

### Accessibility

**Definition**
The ability to access a software function, especially at peak times. A function can be available but not accepted, due to a high volume of requests. Accessibility can be represented by the following formula [61]:

**Measure [61]:**

\[ \text{Accessibility} = \text{time frame to access the service or function} \]
Security

Definition
Ensuring freedom from danger, risk and doubt [50]. It represents a measure of trustworthiness, with security mechanisms such as authentication, authorization, confidentiality, accountability, encryption, auditability, and non-repudiation [58].

Usability

Definition
The quality that emphasizes how a user perceives and interacts with software systems. It is often referred to as the user interface and ease of use [64]. Usability can be defined with the following measures [22]:

Measures [22]:
- Customer preferences
- Application's "friendlier" user interface
- Ease of data reporting and extracting
- Application interoperability
- Application scalability.

Some of the above qualities have quantitative measures, and some rely on qualitative properties. Those with quantitative measures can be represented by numerical values, and include availability, accessibility, time and reliability. These qualities are often related to the network and computational resources. On the other hand, security and usability are qualities with qualitative properties, mainly based on customers’ experiences in interacting with the provided services.

2.4 QR Challenging Features

Despite the popularity of the product-oriented approach to QR, it is still difficult to address the QR of a software system because of their challenging features. For example, QR are multi-faceted entities with different concerns and meanings. They often mean different things for different stakeholders, different requirement views and different domains. An illustration is the identification of time in a radiology service: the radiologists’ view involves less reporting time, whereas the patients’ view is the shortest time before an appointment. Seven QR features have been identified from the literature [7] [17] [18] [6] summarized as follows:
2.4.1 Intangible and Functionally Dependent

High-level problem description and goal specification are the common starting point for software development, that is what the software needs to do and its level of performance. Despite the importance of how well the software needs to perform the job, the ‘how’ question depends on the ‘what’ question. For example, having a swift response time as the main feature in developing software is not a complete requirement without specifying what the software needs to perform swiftly. This places more emphasis on FRs than QR and may lead to implicit or vague definition of QR [7].

2.4.2 Subjective Requirements

Different people with different skills and different interests are involved in defining software requirements. QR/NFRs can be subjective, since they can be viewed, interpreted and evaluated differently by different people [7]. For example, performance can be viewed by a software engineer as minimum processing time, by a system architect as minimum memory consumption, and by a business expert as serving more customers. Stating NFRs imprecisely compounds subjectivity issues throughout software development and evaluation.

2.4.3 Relative and Application-Specific Requirements

NFRs can be relative, since their interpretation, prioritization and evaluation may vary according to the software application and its domain [7] [17], i.e. their fulfilment requires a specific configuration related to their application. Therefore, QR for a specific solution may not be easily reused or generalized to other solutions. This can also be an issue among different functions of a system. For example, in an online shopping system, the security requirement is defined by authentication and encryption and is a high-priority requirement; this is not the case in a catalogue service.

2.4.4 Interconnected and Interrelated Requirements

QR are often interrelated, in that achieving one QR can positively or negatively affect the achievement of another QR [7]. For example, achieving a complex security feature negatively affects the response time of the same function. The scope of a QR's effect can expand to system level; for example, a delay in a single function may cause delay in the overall system. Therefore, trade-off and synergy attributes for prioritization need to be considered in addressing QR. Furthermore, QR realization requires collective and coordinated behaviour of system components and a system-level design strategy [17].

40
2.4.5 Complex Requirements

QR at a high level may be realized by number of functions [6] or a number of QR characteristics at the lower levels (sub-systems or components). However, QR can not be assigned directly to individual system components, but need to be considered at the design of the infrastructure of the whole system before they can be entrusted to system components [17].

Another form of complexity is difficulty in distinguishing between FRs and QR [18]; not all QR can be transformed into functional or empirical requirements. For example, skill level and adaptable user interface are subjective QR; adaptability could be transformed into FR such as language, and follow a standard navigation structure. In contrast, easy-to-learn or stylistic features can not be transformed into functional or empirical requirements. These examples show that it is not always possible to transform a QR into empirical functional requirements, which makes defining all variables that satisfy the QR concerns complex to achieve [18].

2.4.6 Cross-disciplinary Requirements

Throughout the RE process, more QR details gradually evolve, and some QR refinement might require multi-disciplinary knowledge and the involvement of a number of experts [7]. These details can be related to the technical setting or the business domain. For example, a secure payment system involve authentication, authorization, encryption, information confidentiality, network security and other specialized features. In this example, the refinement of secure payment requires a particular level of knowledge and expertise related to security.

2.4.7 NFRs Evaluation

FRs are empirical; functional behaviour can be easily observed and determined whether they satisfy benchmarks [18]. Although some QR are empirical requirements, many, such as reliability and performance, might not be easily evaluated as FRs. QR are runtime concerns and satisfying them in a system means designing runtime mechanisms that fulfil them at the execution time [17]. A quantitative evaluation of QR requires consideration of several runtime factors, such as a target value or range of values in normal, peak, or hazard conditions, for example, demanding fast response during peak time. Moreover, several qualifications may be demanded relating to dependability, subjectivity, relativity, interrelation and complexity of QR. Therefore, QR might not be absolutely achieved and clear cut; they may simply be satisfied sufficiently ('satisficed') [7].
Furthermore, quantitative evaluation can only measure some aspects of QR under specific conditions, which it is difficult to generalize for all system circumstances.

2.5 Issues with Quality Classification Models

Nine of the best known quality models were briefly introduced in this chapter. These models offer a product-oriented approach to QR with hierarchical classifications of quality characteristics and measures. The first seven models are generic to software systems development and the last two are related to a specific type of software system (outsourcing and web application). The hierarchical classification can be divided into two groups. The first classifies quality characteristics into two or three levels of type and sub-type for defining quality measures: ISO/IEC 9126, Roman's NFR taxonomy, Boehm's quality model, FURPS and SERVQUAL. The second identifies software product types or software properties at the highest level of the hierarchy then classifies quality characteristics into several levels down to identifying quality measures: McCall’s quality model, Sommerville’s quality model, FURPS+, Dromey and the quality framework for BPO. Despite the popularity of this approach to address QR, it suffers from the following limitations:

1. It is hard to identify an accepted definition or classification schema for QR, as diverse sets of quality model, taxonomy and agreed standards are available. There is no evidence that any taxonomy is complete or better adapted to accommodate all QR for different types of software system, different application domains and different situations. Different categorization and definitions of QR can lead to confusion over the existing terminology. For example, availability and accessibility are classified as sub-characteristics of reliability in the BPO quality model, whereas in Boehm’s model they are sub-characteristics of efficiency. Some quality characteristics can be found at different levels of the hierarchy. For example, efficiency appears in the highest level in the ISO/IEC 9126 model, McCall’s and Roman's taxonomy, whereas it appears at a lower level in Boehm's model, as a sub-characteristic of as-is utility, and in FURPS as a sub-characteristic of performance. Another example is the security characteristic, which appears in the highest level in the ISO/IEC 9126 model but as a sub-characteristic of reliability in the FURPS model.

2. Some 227 distinct QR can be found in the literature, listed in Appendix A. Each can be defined differently depending on the application type, domain and conditions. Furthermore, different people use different terminology, which makes using a given
generic classification and definition difficult. As shown by the nature of QR (particularly the intangible, functionally dependent and relative and subjective characteristics) such generic, abstract and out-of-context definitions can produce only subjective definitions of QR. Therefore, the above models can offer no more than a generic reference or catalogue that guides capturing and defining QR of software systems.

3. The third limitation is related to the second, the definition of generic QR. It is difficult to identify a generic QR definition without a supporting process. The classification models lack this process support as shown by the nature of QR (particularly their interrelated, complex and cross-disciplinary characteristics), it is difficult to address any generic or abstract definition of QR without a supporting process.

2.6 Summary

This chapter aimed to identify the QR of software systems and also highlighted the current issues of QR models that hinder the advancement of QR modelling. It introduced quality concepts of software systems, and then in section 2.2 reviewed nine quality classification models. Section 2.3 offered definitions of the QR used in this thesis. It is important to note that we are not proposing a generic definition of QR; these definitions are used only as a reference to provide consistency while explaining and evaluating the proposed framework. Section 2.4 provides a summary of the QR’ natural features, justifying the difficulty of addressing QR. Finally, section 2.5 highlights some issues related to the QR classification models. summarized in the following three points:

- None of the classification models addresses a complete list of QR.
- All the models offer generic definitions of QR which are subjective, as the definition may differ according to the application type, software domain and environment conditions.
- The classification models do not provide detailed guidelines for a methodology for adopting the proposed models; as a result, customizing the suggested generic definition of QR is difficult.

The following chapter introduces requirement modelling frameworks to analyze their ability to address the above limitations, and also to analyze the extent to which the QR’ natural characteristics QR are considered.
3 Requirement Modelling Framework

3.1 Introduction

The success of a software system is measured by the degree to which it meets the purpose of its development. Therefore, identifying the purpose is one of the key activities in software system development. This is the main objective of requirements engineering (RE), a branch of software engineering first recognized by Alford in 1977 [65]. Since then, numerous academic and industry researchers have emphasized the essential contribution of RE to the success and quality of software products. The scope of the RE discipline has been extended to include a variety of concepts, processes, approaches, techniques, tools, methods and frameworks. This chapter reviews the areas of RE that influenced the design of the QRMF. Section 3.2 reviews basic concepts in RE and requirements processes; it also highlights the development features throughout the requirements process. Section 3.3 provides a detailed review of the goal-oriented requirements modelling approach. It includes a detailed overview of the most popular requirements modelling frameworks that address the QR. A brief overview of other requirements modelling approaches is given in section 3.4. The issues related to QR classification models are highlighted in section 3.5. Finally, a summary of the chapter is provided in section 3.6.

3.2 Requirements Engineering and Requirements Engineering Processes

One of the oldest definitions of RE is "requirement definition is a careful assessment of the need that a system is to fulfil. It must say why a system is needed, based on current or foreseen conditions, which may be internal operations or external market. It must say what system features will serve and satisfy this context. And it must say how the system is to be constructed" [66]. This definition includes most of the concepts addressed by RE, including the goals that specify why a software system is needed, the functionality to achieve these goals, the agents and resources that characterize the current domain features and expected environment conditions to realize the goal, and constraints on how the software must be designed and implemented. Ten years later a more comprehensive RE definition was introduced: "Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behaviour, and to their evolution over time and across software families" [40]. Besides defining the concepts addressed in RE, this definition refers to the importance of precise specifications which provide a basis for requirements analysis,
validation and verification. The definition also refers to the evolution of requirements over time and across software families, emphasizing the important of adapting to changes in the real world and the possible reuse of the specifications. Recently, Lamsweerde described RE as the process of discovering, understanding, formulating, analyzing and agreeing on what problem should be solved, why such a problem needed to be solved, and who should be involved in the responsibility of solving the problem [67]. These definitions reflect the nature of RE as a multi-disciplinary and human-centred process, as the whole software development cycle and its operation are related not only to technical issues and problems but also to managerial, organizational, economic, and social issues.

RE plays a critical role in the software engineering life cycle. Research has provided practical evidence that failures in software projects are by and large due to poor requirements specification [1]. Poor requirements mean a poor foundation for the software project, which leads to project failure or solving defects at increased cost and delay [8]. Furthermore, those projects surviving with poor requirements specification and documentation will face difficulties in adapting to changes or upgrades [68].

The RE community has classified software requirements into functional and non-functional requirements. The functional requirement specifies what the software will do [41]. This includes the required behaviour of the system in terms of required activities or services, the input and output variables of these activities, the states of the variables before and after commencing an activity, the conditions in which these activities take place, and the scope of the software [68]. On the other hand, the non-functional requirements (NFR) or quality requirement (QR) describe how well the software systems accomplish the requirements [41], distinguishing the acceptable level of the software product. NFRs describe the quality characteristics that the software solution must satisfy and the design and environmental constraints related to the software system. NFRs are generally more difficult to express and analyze in an objective and measurable way because of their intangible nature.

The first stage of software development is to identify the problems needing to be solved. This seems to be straightforward at a first glance, as customers presumably know what they need. However, defining the right problem can be surprisingly difficult. The reality of software development practices reveals that customers frequently do not know the real problems they want to solve [69]. Therefore, together with software analysts, they need to discover, understand, formulate, analyze and communicate the problems to be
solved. These activities take place at the start of the software development process, known as the RE process. This process primarily concerns understanding the problem space [70] that initiates the demand for developing or enhancing a system. This involves understanding the current system and technical context to acquire an understanding of the problem and its causes. A precise definition of the problems that the software is intended to solve determines what the software is intended to do. Many RE processes have been introduced over the past two decades [38, 41, 68, 71-74], each process having its own merits, different granularity and focusing on different aspects of RE.

A review of RE processes introduced in the literature revealed a total of seven different activities. These are activities with a distinct task, i.e. if the same task is described by two different activity names in different processes we include it under the most common name; for example, documentation, modelling and specification activities are referred to as documentation activity. The seven activities are: domain understanding; elicitation; analysis and negotiation; verification and validation; modelling and documentation; management; and quality assurance. The sequence among these activities can not be described as clear cut as most of them are interrelated and intertwined, and backtracking between activities may be required to check the consistency of requirement specifications. Furthermore, each of these activities can be considered as a process in itself, that includes a set of activities to perform certain goals.

Although the development of knowledge throughout the RE process influences the design of the QRMF proposed in this thesis, it is beyond the scope of this research to discuss all the RE processes found in the literature. However, the remaining part of this section provides a detailed explanation of the core RE activities outlined in [41], and how the knowledge evolved throughout these activities relates to the design of the framework.

3.2.1 Requirements Modelling, Specification and Management

Normally requirements modelling, documentation, specification and management are described as individual activities in the RE process, as each of them is used for different purpose. However, in this research we consider these activities as tools or techniques that can be used separately or together to facilitate and support the other RE activities, which are the core for developing requirements knowledge. The requirements modelling plays a fundamental role in the RE process [38] as it delivers a structured description of requirements that can be used throughout the requirements process and the rest of the software development processes. In addition, it is a main communication
language between stakeholders with different backgrounds [70]. Modelling notations provide an abstract level of requirements description by providing a vocabulary and structural rules for the problem components [70]. Modelling helps to analyze the requirements and identify detailed requirements [38].

The wide range of modelling techniques and notations found in the RE literature have been developed by academics and industry. They range from informal models for early requirements to more formal models for late requirements that can guide software developers [70]. There are three types of modelling language: informal, semi-formal and formal. Informal language is based on natural language; it is easy to use and understand but results in ambiguity and inconsistency [1]. Semi-formal language has a formal syntax but informal semantics; the most widely used include the Unified Modelling Language (UML) [75], BPMN [75] and goal-based techniques [76]. Formal language is typically based on a mathematical foundation such as logic or algebra; it has a well-defined formal syntax and semantics, such as Specification and Description Language (SDL) [77]. Some of the formal notation is supported with graphical notation to enhance its readability and comprehensibility, such as Petri Nets [78] and Object Constraint Language (OCL) [79].

The requirements specification, also called requirements documentation, is the process of documenting and structuring the agreed requirements using a modelling language. Specifying the requirements in different types of language facilitates the readability and understandability by requirements engineers and customers, and also facilitates retrieval and traceability for analyzing, validating and managing the requirements. Most RE processes position documentation after the analysis and negotiation activities [73] [80] [68], but in reality documentation as well as modelling should start at the beginning of the RE process [1]. Setting up a blueprint for documentation at the very start can help documentation process while requirements are evolving. This also enables requirements management to be handled throughout the RE process.

Requirements management is also an ongoing activity, with requirements evolving over time and across product families. As requirements evolve, tracking and updating the documented requirements and models are required. This can be challenging as modification to one requirement may affect others, demanding consideration of trade-offs between cost and benefits and re-evaluation of the updates and overall documents. This challenge is increased when organizing a large number of requirements or distributed
requirements [41]. Tools and techniques have been developed to ease, and automate, the identifying and documenting traceability links among requirements. These techniques may include traceability links of possible change impacts, version control [81], managing consistence [81], software product families [82], and stability analysis [81].

The use of the tools and techniques throughout requirements development facilitates constructing the quality requirement output. This final output of RE is called the Software Requirements Specification (SRS) document, defined as “A document that clearly and precisely describes each of the essential requirements (functions, performance, design constraints, and quality attributes) of the software and external interfaces. Each requirement is defined in such a way that its achievement can be objectively verified by a prescribed method, for example, inspection, demonstration, analysis, or test” [83]. A good SRS document exhibits the following characteristics [84]:

- **Unambiguous**: SRS is unambiguous if every requirement has only one interpretation.
- **Complete**: SRS is complete if all the requirements are included, defined, labelled and comply with pre-defined standards.
- **Verifiable**: SRS is verifiable if every requirement specified can be verified against a predefined checklist or international standards.
- **Consistent**: SRS is consistent if no requirement described is in conflict with another.
- **Modifiable**: SRS is modifiable if any necessary changes can be made easily, completely, and consistently.
- **Traceable**: SRS is traceable if the origin and rationale among requirements is clearly stated, and forward and backward links between requirements can be traced.
- **Usable during the operation and maintenance**: SRS must address the needs of the operation and maintenance.

### 3.2.2 Requirements Elicitation

This activity is preliminary to capturing knowledge about the current system and the customer’s requirements. Lamsweerde [80] separates it into two intertwined processes: domain understanding and requirements elicitation. Domain understanding refers to understanding the current system and the technical context which leads to understanding the problem and its causes. It includes knowledge about the organization,
the users involved, domain constraints and the current system. Requirements elicitation refers to discovering the candidate requirements, the scope of the system and the assumptions that will shape the solution. Elicitation involves exploring the problem with stakeholders to identify the real problems and their causes, as well as exploring opportunities and alternatives to the new system. This knowledge is by no means available when the project starts, as normally only high-level information is available at this point [38]. In most projects, requirement engineers lack detailed knowledge about the business domain and stakeholders lack a clear understanding of what they want [85]. A gradual learning curve for both the stakeholders and requirements engineers when working together reveals requirement details. This is one of the reasons for the common observation about the tendency of requirements elicitation activity as a messy process [73]. The output of the elicitation process is largely unstructured and may contain much irrelevant, contradictory, inconsistent and ambiguous information which must be refined in later stages.

Nevertheless, this activity is a very important base for the process of RE and the software development cycle as a whole. The use of the right techniques to elicit knowledge and remove communication barriers between various stakeholders is one of the most common challenges during this activity. Techniques are based either on different forms of interaction with stakeholders, such as interview, observation, focus groups and brainstorming, or on artefacts to help acquire relevant information, such as scenarios, prototype and knowledge reuse. The output of this activity is the preliminary proposal of the contextual description of these aspects, will be expanded throughout RE activities. Another important output is a glossary of terms which will be used throughout and beyond the RE process to ensure consistency.

3.2.3 Requirements Analysis and Negotiation

Requirements analysis is an essential activity as it is the foundation for requirements implementation decisions. Requirement engineers take the lead in this activity, to refine the output of requirements elicitation (the preliminary draft of the requirements document) in a more structured manner. The requirements are modelled and analyzed in order, first, to help understand the elicited requirements thoroughly; secondly, to identify irrelevance, ambiguity and incompleteness in the elicited requirements; thirdly, to manage conflicting requirements through stakeholder negotiation, prioritizing their concerns and evaluating alternatives; fourthly, to explore potential risks (safety hazards, security threats, development or performance risks) which
need to be carefully analyzed to identify when they are likely to happen, what is their critical level, and how to mitigate them in a realistic and robust manner; and finally, to develop requirements test cases for important functions, to be used for testing the system. These five objectives can be complex because they are interrelated, e.g. addressing one aspect may result in changes in other aspects. Therefore, iterative checking for inconsistency and conflict and evaluation of alternatives and risks, are required.

Requirements analysis and negotiation are preformed with the support of modelling languages. Tools and techniques vary in terms of complexity (checklist [36], simulation [37]), and formalization (prioritization[86], ontology[87], formal methods [88]). The selection of a suitable analysis technique depends on the modelling notation used for requirement description. One of the key issues in this activity is selecting the appropriate notation to model requirements at the right level of detail [1]. The output of the process is a more structured, consistent and complete set of requirements than the elicitation output.

3.2.4 Requirements Verification and Validation

This activity examines the quality of the constructed SRS and ensures satisfying stakeholders’ requirements. Pfleeger [68] distinguished between verification and validation: verification ensures building the system right and validation ensures building the right system. Verification is checking that the constructed documents and artefacts do not conflict with each other, and also conform to the requirements definition. On the other hand, validation is checking that the defined requirements match customers’ expectations and address potential risks. In other words, identified artefacts must accurately express stakeholders’ needs [41]. This activity requires direct involvement from stakeholders as they can verify the identified models and documents against real-world problems [41]; this implies their agreement to the identified requirements. However, it is not a simple activity, needing expertise and awareness of social concerns. Knowledge and expertise about the domain are required to ensure that the problems are correctly captured. The social concerns are demonstrated in negotiation and resolving conflict between stakeholders’ requirements without weakening their goals [38].

Requirements verification and validation techniques vary from simply reading the documents and reporting errors to formal requirements inspection. Details of requirements verification and validation techniques can be found in (13) (7) (7, 13). As stakeholders are involved, validation techniques are usually visual and less abstract,
demonstrating real-word problems; they include prototyping [89], animation [90], simulations [37] and scenarios [42]. Formal evaluation, such as ontology validation, can be applied to formal quality models to enable or ease verification [91].

It might appear that the validation and verification stage is simply repeating the requirements analysis and negotiation stage. However, these are two different stages, each employing different techniques and producing different outputs. Confusion may occur because both stages aim for the same quality attributes. These generic quality attributes are needed at both the software development stages, although each stage addresses different views; for example, the requirements analysis and negotiation stage addresses the customers' view of requirements, while the verification and validation stage addresses a system view of requirements. The output of the requirements analysis and negotiation stage is the input to the verification and validation stage, which is finalized in the SRS document; this, in turn, is the input to the following stages of the software engineering process [1].

3.2.5 Requirement Development Throughout the Requirement Process

The previous section detailed the various activities in the requirement process. These activities are interrelated, iterative, and may span the entire software system's development life cycle [38]. Although no single process can be considered as a standard RE process, in all the processes the requirement knowledge matures gradually in parallel with the progress of the processes. In the other words, a requirement process normally starts with poorly defined and conflicting demands which gradually evolve until a mature understanding of actual requirements and their context is reached [68, 73, 80]. Throughout the process, different levels of detail are revealed, evolving from abstract descriptions of business requirements to detailed technical requirements relevant to the system design. At each level, various models are used to express the requirements at the appropriate level of detail and in semantics understood by the stakeholders involved. This behaviour of requirement development is described as the early and late requirement phases [92].

The early phase of requirements is about discovering the purpose of the project by understanding the motivation and rationale that underlie the demand for a system. This phase also focuses on capturing the organizational context, the stakeholders involved and their relationships. During this stage it is important to capture all stakeholders' demands and possible ways to address them [92-94]. Not all the information captured at this phase
will be used for developing the system [95] as conflicting, irrelevant, incomplete and ambiguous information can be captured [93]. However, it is important to capture all possible information to provide a deep understanding of the enterprise setting, construct the correct requirements and provide precise justifications for each them [92]. This phase is mainly expressed through informal or semi-formal language, to be understood and revised by the stakeholders involved at this stage.

In contrast, the late requirement phase concerns capturing the system features and the operational environment. This includes refining requirements until operationalization, defining related system components [93] and capturing related design and environment constraints [94]. The agreed requirements and related constraints are expressed formally or semi-formally in this phase, as the focus is on producing an adequately specified requirements document [92]. The transformation of the business requirements captured in the early phase to the technical requirements captured in the later phase is conducted with the aid of requirements modelling frameworks. The following section reviews some of the better known frameworks.

3.3 Goal-Oriented Requirements Modelling

A goal is a non-operational objective that motivates the introduction of a new system [96]. Lamsweerde [72] defines a goal as an objective to be achieved through agents and the environment of the system under development. Antón [97] defines a goal as the high-level objective of a business, organization or system that captures why a system is needed and guides organizational decisions [97]. Goal-Oriented Requirement Modelling (GORM) uses goals for eliciting, elaborating, structuring, specifying, analyzing, negotiating, documenting and modifying requirements. It views the system under development and its operating environment as a collection of agents (humans or system components) with assigned responsibility to achieve goals. GORM can be described as a process to identify requirements specifications through capturing stakeholders’ demands and the rationale for those demands. This process involves agent-based reasoning through assignment of responsibilities for goals and constraints among agents. Furthermore, it supports the RE process with explicit modelling, refinement, and operational analysis of goals [98]. Goal-based models can be used as excellent communication tools among decision makers as they offer the right level of abstraction for choosing or suggesting alternative operations.

This approach is becoming increasingly popular as it is the only one to capture the rationale for developing a software system. GORM takes a wider systems engineering
perspective, which provides added support for the early requirements analysis. Capturing the rationale helps to understand the requirements with respect to high-level concerns in the problem domain. This is really important, as higher-level concerns, i.e. goals, are more stable than lower-level concepts, i.e. requirements, processes and data [8] [97]. This encourages the start of the RE process, as GORM ends where other RE approaches begin [98]. It focuses on capturing high-level concepts where other approaches focus on the formulation of the software system requirements.

One of the key features of GORM that is especially relevant to this research is its treatment of QR. GORM is the only RE approach that rigorously treats QR through modelling, refinement, operation, and analysis of the operational alternatives. The QR are represented by softgoals with no clear-cut satisfaction condition. High-level QR are abundant in organizations, and quite frequently the success of a system depends on satisfying them [98]. The core GORM approaches are briefly described in the remainder of this section.

3.3.1 The NFR Framework

The NFR Framework constitutes a goal-oriented RE approach to capturing QR. Chung introduced the NFR Framework in [99] then further developed it in [19]. The NFR Framework aims to model and analyze QR through capturing the domain NFRs then decomposing them, identifying possible operationalization, dealing with ambiguities, trade-offs, priorities, and interdependencies among them. It offers sufficient visibility to capture all relevant NFRs and their interdependencies, and helps designers to understand the necessary actions to address the QR of the software system. Furthermore, the framework supports selection decisions among operational alternatives with design rationale, and evaluating the impact of decisions. The main purpose of this approach is to systematically model and refine NFRs and to expose positive and negative influences of different operational alternatives.

The NFR Framework offers a catalogue of NFRs and associated development techniques to provide a terminology and classification of NFR concepts. Figure 9 shows a catalogue of some of the NFR types; those in bold face are separately detailed in [19]. The NFRs are arranged in a hierarchical classification, with general NFRs at a higher level and specific ones at lower. For example, performance is a generic type with time and space as sub-types, which in turn have their own sub-types. The standard development techniques along with methods of decomposing NFRs are organized into method catalogues. An
example of a method catalogue for confidentiality is shown in Figure 10. These catalogues can be accessed by system developers from early requirement elicitation to aid capturing QR and operational alternatives.

![Figure 9 A Catalogue of Some NFR Types [19]](image)

![Figure 10 Method Catalogue of Confidentiality [19]](image)

The NFR Framework captures and analyzes QR on a Softgoal Interdependency Graph (SIG) which represents and interconnects the QR for the system under development. The SIG is a tree structure diagram that facilitates capturing and interconnecting abstract softgoals with different levels of refined softgoals until reaching operational softgoals. A softgoal is defined in terms of Type and Topic as “Type [Topic]” e.g. reliable [reporting]. Type is a non-functional concern while Topic defines the context for the Type; when put together they represent a world phenomenon [100]. The framework supports three types of softgoals: NFR softgoals, operational softgoals, and claim softgoals. The NFR softgoals represent the QR to be considered. The operational softgoals are techniques to satisfice the softgoals. The claim softgoals allow the analyst to record design rationale for
softgoal refinement, softgoal prioritization, softgoal contribution, etc. An example of the SIG shown in Figure 11.

The softgoals can be refined to more specific sub-goals at several levels using AND or OR refinement links. This refinement can last until alternative solutions are reached. Also, the softgoal contributions can be captured with positive ("+") or negative ("−") contribution links. The contribution link indicates how well the solution contributes to the goal achievement where Make (denoted by++) indicates a good enough solution while Hurt (denoted by--) indicates a bad solution. Table 1 illustrates the SIG contribution types.

Table 1 SIG Contribution Type

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Contribution</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>++</td>
<td>MAKE</td>
<td>Child label is strongly propagated to parent.</td>
</tr>
<tr>
<td>+</td>
<td>HELP</td>
<td>Child label is somewhat propagating to parent.</td>
</tr>
<tr>
<td>=</td>
<td>EQUAL</td>
<td>The two softgoals share the same label.</td>
</tr>
<tr>
<td>-</td>
<td>HURT</td>
<td>Negated child label somewhat propagating to parent.</td>
</tr>
<tr>
<td>--</td>
<td>BREAK</td>
<td>Child label strongly is negated and propagated to parent</td>
</tr>
<tr>
<td>?</td>
<td>UNKNOWN</td>
<td>Interdependency unknown, child does not affect to parent.</td>
</tr>
</tbody>
</table>

Figure 11 Softgoal Interdependency Graph [19]
By analyzing the alternative operations, with the aid of the above contribution types, one best operation can be selected to meet high-level QR, and therefore the set of selected leaf-level softgoals can be implemented in the software. The softgoals are also annotated with a status label which indicate the status of softgoals and operationalizations. The label typically annotates the selection decision of method or technique for operationalization. For the more abstract softgoals, the label describes the status of all the other NFRs. Table 2 show different status of soft goals.

### Table 2 Softgoal Status Labels

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>Satisfied</td>
<td>Softgoal is fulfilled or chosen to be implemented.</td>
</tr>
<tr>
<td>w+</td>
<td>Weakly satisfied</td>
<td>There is some positive support to the softgoal.</td>
</tr>
<tr>
<td>U</td>
<td>Undecided</td>
<td>Realization of the softgoal neither confirmed nor denied</td>
</tr>
<tr>
<td>w-</td>
<td>Weakly denied</td>
<td>There are some indicators against fulfilling the softgoal</td>
</tr>
<tr>
<td>X</td>
<td>Denied</td>
<td>Softgoal can not be realized or is chosen not to be implemented</td>
</tr>
<tr>
<td>Lightning</td>
<td>Conflict</td>
<td>There are conflicting contributions to this softgoal. Some supporting, some against</td>
</tr>
</tbody>
</table>

#### 3.3.2 i* Framework

Yu introduced the i* framework in [14] as a strategic modelling framework for RE. In the i* framework, goals are defined based on agents' demands, and this tied relationship between goals and agents means that i* can be described as a goal-oriented or agent-oriented modelling framework. i* can be used for both the early and late phases as it supports the modelling activities before and after formulating system requirements. At the early requirements stage, it helps to understand why a new system is needed. On the other hand, at the late stage it helps to propose the new system configurations and processes, and evaluate the extent to which the FRs and NFRs meet the user's needs. The framework has two main components: the Strategic Dependency (SD) model and the Strategic Rationale (SR) model.

The SD model provides a network of external dependency relationships among actors, also called social dependencies. The dependency is a relationship between two actors, a depender who relies on a dependee to accomplish an intentional goal. The dependency relationship is characterized by intentional elements called dependa. The primary intentional elements are: resource, task, goal and softgoal. These are the basic elements of an organization. A softgoal represents the QR or the goals that can be partially
satisfied. The SD model captures abstract intentionality of the processes in the organization and what is important to its participants. This model facilitates analysis of the direct or indirect dependencies among agents and also explores their opportunities and vulnerabilities. The SD model offers four types of dependency based on the type of dependum: goal dependency, resource dependency, task dependency and softgoal dependency. The model also distinguishes the dependency strength: an open dependency refers to failure of the dependum to affect the depender’s goals to some extent; a committed dependency is the failure of the dependum significantly to affect the depender’s goals; and in a critical dependency the dependum’s failure seriously affects the depender’s goals, and all courses of action will fail. Figure 12 shows the example of an SD model for a healthcare configuration.

![Figure 12 Strategic Dependency Model Example](image)

The SR model explicitly explores the rationale behind the processes, in terms of process elements and relationships among them. The model provides a lower-level abstraction and deeper understanding of the intentional aspects of an organization and a system. It also enables analysis of possible alternatives in the definition of the processes to better address actors’ concerns, through its ability to analyze in great detail the internal processes within each actor. As with the SD model, the SR model offers four types of node and the same dependum types. The SR process elements are linked through decomposition and means-ends relationships. The decomposition relationship is used to model AND/OR refinement to lower levels. The means-ends relationship is used to specify
alternative ways to achieve goals. A means-ends link indicates the means by which a goal can be achieved, a task accomplished, a resource produced or a softgoal satisficed. This relationship also specifies positive ("+" and "++") and negative ("–" and "––") contributions to the softgoals. The softgoals are used as selection criteria to decide among alternative configurations to meet the system requirements. Figure 13 shows the example of an SR model for an insurance claims configuration.

Figure 13 A Strategic Rationale Model Example [14]

Tropos is a requirements-driven agent-oriented development methodology that is based on the i* framework [101]. Tropos guides the development of agent-based systems from the early requirements phase to implementation. Tropos has an associated formal specification language to capture the domain’s semantics, including constraints, invariants, pre- and post-conditions, in the i* notation [102].

This modelling framework is also described as Agent-Oriented Requirement Modelling (AORM) [14] [103]. Agents play a central role in identifying agent intentions, symbolized by goals and softgoals; agent responsibilities are symbolized by action; objects required by agents to perform an action are symbolized by resources; and finally the relationships between agents are symbolized by action dependency links.
3.3.3 KAOS

KAOS was introduced in 1990; it stands for Knowledge Acquisition in Automated Specification of Software [104] and also for Keep All Objectives Satisfied [105]. It is a well-developed methodology in the GORM approach, with rich formal analysis techniques. KAOS offers a multi-modelling paradigm which combines different levels of expression and reasoning, including semi-formal methods to model and structure goals, qualitative means of selection among alternatives, and formal modelling for accurate reasoning [105] [67]. Each construct has a two-level structure: an outer graphical semantic layer for concept declaration including attributes and relationships, and an inner formal layer for formal definition of concepts.

KAOS offers a top-down requirements elicitation process starting from abstract goals and refining them to the operational level. The KAOS multi-modelling paradigm integrates the following core modelling views, as shown in Figure 14:

- The goal model structures the collected goals into directed, acyclic graphs where each goal is represented, justified, refined, operationalized and assigned to agents. The goal model also identifies obstacles and conflicting goals, e.g. a system state that cannot satisfy two goals simultaneously, and their mitigation. This view justifies why a system is needed, and the goal is refined through the answer to how to achieve it.

![Figure 14 KAOS Notation and Concepts](image-url)
• The responsibility model expresses agents (human or automated components) that are responsible for achieving requirements and expectations. This model is closely related to the goal model, but goals are no longer refined when placed under the responsibility of a single agent. This view represents who in the system takes a part in fulfilling the specified need.

• The object model expresses the system structure through identifying objects related to domain concepts or operationalizing requirements with their attributes and relationships. This view refers to which data objects need to be addressed to perform system functions.

• The operation model expresses the composition of services and operations performed by each agent to satisfy requirements and related goals. The compositions are made through data flows (input and output) or control flow (events trigger or stop another operation). This view refers to what will perform system functions and when.

Goals in KAOS may refer to services to be delivered and the QR to be achieved. The goals are organized in an AND/OR hierarchal refinement graph where goals are refined until assigned to an individual agent. At the agent level, requirements are a goal under the responsibility of an agent and the expectations are a goal under the responsibility of an agent in the environment. KAOS addresses both functional goals and non-functional goals; however, its representation does not differentiate between them.

3.4 Other Requirements Modelling Approaches

The RE literature offers a wide range of requirement modelling approaches, such as goal-oriented [98], problem-oriented [106], aspect-oriented [107], service-oriented [108], object-oriented [109] and process-oriented RE [110]. This section offers a brief overview of the two most commonly used modelling approaches, which are also used to evaluate the QRMF features.

3.4.1 Object-oriented Requirements Modelling

The object-oriented concept was adopted by software developers in the 1960’s, with the development of the Simula programming language [109]. Since then, the object oriented paradigm has grown in popularity, adopted by different aspects of software development such as programming language, software design, RE, etc. The object-oriented approach is based on encapsulation of the satiable concepts as objects, characterizing their behaviours, their inter-relationships and reusing them in different software settings. Well
defined structures for system components and application settings enable long-term benefits such as enhanced productivity, reduced maintenance costs, improved consistency and better integration of software components. Therefore, it has been widely adopted in complex and large-scale information system [111].

Object-oriented modelling encapsulates information about the process, product and functionality of the domain into requirements objects [112]. This is conducted through encapsulating the actions, data and behaviour within a new requirement object; and also through creating specialized objects based on specified behaviour of the existing objects. Specialization, also called generalization, organizes the requirement objects into hierarchies that reflect different kinds of requirements and their relations. Such a structured treatment of requirements facilitates reuse of domain-level objects in various settings of large complex systems and in high-level system design [113]. Another feature of object-oriented modelling is that the requirements objects can formulate semantics for communication between stakeholders, software engineers and project managers. There are several modelling notations for object-oriented modelling, such as Data Flow Diagrams (DFD), and Entity Relationship (ER) and Unified Modelling Language (UML) diagrams. UML is the most popular modelling notation, offering different requirement modelling views; it has been adopted in this research to demonstrate some of the framework features. Further details on UML can be found in [75].

3.4.2 Business Process Requirements Modelling

The business process modelling approach aims to formalize business processes in the organizational context of their execution. Business process models are used for presentation, identification, validation, improvement and implementation of process-oriented systems [110]. The approach facilitates capture of the dynamic behaviour of a model, to be understood by both stakeholders and information system developers. A business process model documents and communicates business activities through visual modelling and also allows formalizing the computational support to these activities through formal modelling. The visual modelling, such as using BPMN [114], DFD and UML activity diagrams, is generally regarded as easy to understand by humans. The formalism, such as WSBPEL [115] and Petri Nets [78], while providing software execution details, are difficult to understand by humans.

Business process-oriented requirements modelling (BPORM) uses the business process modelling at the start of the requirements process to derive alternative sets of
requirements for a process-oriented system [116] or service-oriented system [108]. These systems are large, complex and dynamic, i.e. the stakeholders’ requirements are often changed [113]. It is very difficult to anticipate the future requirements of such systems at an early stage. However, with the process-driven RE approach, business processes can be flexibly structured and shared across the different sectors of a large organization [113].

3.5 Issues Related to QR Modelling in Existing Requirement Modelling Approaches

Quality requirements modelling frameworks overcome the limitations of the QR classification models introduced in the previous chapter, by offering a process-oriented approach that systematically captures the QR of a specific domain, condition and operational environment. Such a systematic and detailed methodology facilitates capturing QR, refining QR them into operational options, selecting among the operational options with the rationale behind the selection and also tracing the evolution of QR through graphical models. Furthermore, most of these approaches use the generic QR classification models as a catalogue to guide the process of customizing the proposed generic definitions to the software under development. For example, the NFR Framework offers detailed classification catalogues of the QR and other catalogues for possible methods to realize them. As a result of this methodology, the difficulties related to the nature of QR are better addressed. For example, the subjective definitions of QR are avoided by capturing the QR with interrelated software and environment conditions and also with different stakeholders’ concerns. This is visible through the SD model in the i* framework. Another example is explicitly capturing QR in the requirements model to identify their intangible characteristics QR and also illustrate the function-related concepts.

Despite the fact that the QR modelling frameworks overcome the limitations of quality classification models, the QR of software systems are still not as well recognized as functional requirements [117]. This research argues that this is because the modelling frameworks described in the literature address QR from only single or limited views of requirements; for example, the systematic process of NFR framework models focuses on the goal view of requirements, and i* focuses only on the goal and agent views. Given the diversity of concepts that need to be modelled, a single model can not represent all requirement views [24] in all phases of requirements. This argument has been introduced to support multi-perspective modelling for functional requirements [118] [119] [120]
[121] [122] [82]. However, QR have a more complex nature than functional requirements. Therefore, we argue that a single model is not sufficient enough to capture a complete set of QR.

Although KAOS captures four different views of requirements (goal, agent, operation and structure), the QR are addressed as softgoals and use the same notation as functional goals. Using the same notation restricts the emphasis on QR. Such an emphasis is important as QR can be overlooked because of their intangible nature. Although QR are more complex and easier to overlook than FRs, none of the approaches explicitly annotates QR in all views of requirements or in integration with FRs models.

The NFR Framework offers the most comprehensive treatment of QR; however, the QR are modelled and analyzed in a separate model which does not capture the environmental features such as the dependency or behavioural relationships of the system.

Requirements modelling is an iterative process [68, 73, 80] where understanding evolves from an abstract description of business requirements in the early phase to detailed technical requirements in the later phase. At each phase, various types of model are used to express requirements, each of which addresses specific requirement views or level of detail. Capturing the details of each stage facilitates traceability among models, which is important in justifying the logic behind the final models and also helps to understand them. This importance was recognized by [120] [121]. Tomoyuki [121] proposed and justified a modelling framework to bridge this gap; however, this solution only addressed functional requirements. Therefore, we argue that the QR modelling approach should construct QR models at both stages and also in the phases that map between them.

<table>
<thead>
<tr>
<th>Modelling Framework</th>
<th>QR modelling issues</th>
<th>Include all requirements views</th>
<th>Explicitly annotate QR</th>
<th>Integrate QR with FRs models</th>
<th>Capture early and late requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFR Framework</td>
<td>Goal</td>
<td>Yes</td>
<td>No</td>
<td>Early-late</td>
<td></td>
</tr>
<tr>
<td>I* Framework</td>
<td>Goal, Agent</td>
<td>Yes</td>
<td>Yes</td>
<td>Early</td>
<td></td>
</tr>
<tr>
<td>KAOS</td>
<td>Goal, Agent, Operation, Structure</td>
<td>No</td>
<td>No</td>
<td>Early-late</td>
<td></td>
</tr>
<tr>
<td>UML</td>
<td>Structure, Operation, Agent, Architecture, Behaviour</td>
<td>No</td>
<td>No</td>
<td>Early-late</td>
<td></td>
</tr>
<tr>
<td>BPMN</td>
<td>Operational, Architecture, Behaviour</td>
<td>No</td>
<td>No</td>
<td>Early</td>
<td></td>
</tr>
</tbody>
</table>

Figure 15 Requirements Modelling Framework and QR Modelling Issues
We observe from the above issues that the requirements modelling frameworks introduced in the literature address some of the above issues, as highlighted in Figure 15. We argue in this research that to provide complete and comprehensive QR models, these four issues need to be addressed in a single framework.

3.6 Summary

This chapter reviewed the concepts, processes and modelling approaches that address the requirements of software systems. The main contribution of this chapter is to highlight the limitations of current RE modelling approaches in addressing QR, the motivation for this research. In order to illustrate that, the chapter started with an overview of the basic concepts of RE, its process and how requirements are developed throughout the process. Understanding how requirements evolve throughout the process is significant because these features can not be missed in the design of the QRMF. The chapter then provided an overview of a goal-oriented modelling framework, focusing on three commonly used frameworks that address QR: NFR Framework, i* Framework and KAOS. Object-oriented and business-process requirements modelling approaches were briefly introduced as their notations are widely used throughout the requirements process. Finally, the chapter highlighted the issues of addressing QR in requirements modelling frameworks. It is important to note that, although these issues are partly addressed in the existing frameworks, addresses all of them. A summary of these issues follows:

- QR are addressed from only single or limited views of requirements.
- QR are addressed either at the early or later stage of requirements, but not at both stages or the mapping between them.
- Limited frameworks capture QR using specific notations.
- Limited frameworks capture, refine and justify QR in integration with FRs including functional dependency and environmental conditions.

This research proposes a multi-perspectives modelling framework, QRMF, that addresses the limitations of both QR classification models (shown in Chapter 2) and QR requirements modelling frameworks (shown in this chapter). Before the framework design is introduced, the following chapter offers a background study on the multi-perspective modelling frameworks that influence the design of QRMF.
4 Integrated Modelling Approaches for Software Systems

4.1 Introduction

The previous section presented different approaches to addressing requirements, each focusing on only one or a few views of requirements. However, all requirements views are important in constructing a complete requirements model. Furthermore, a single model can not capture sufficient details to cover all the different views of requirements. A requirements model may suffer from over-emphasis on one view, overlooking other important views. The requirements model may become crowded with details that hinder understanding of the model. These two factors can lead to an immature requirements model and consequently to software failure. As a remedy, we propose a framework that organizes the description of system requirements using an integrated structure of all requirement views. Several integrated modelling frameworks are described in the literature, such as the Zachman framework [118], Curtis's Modelling Process [119] and Multi-perspectives Business Process Modelling (BPM) [120]. An overview of these frameworks is given in sections 4.2 to 4.4. This is followed by a review of the common features of multi-perspectives modelling frameworks.

4.2 Zachman Framework for Enterprise Architecture

The Zachman framework for enterprise architecture [123] established a fundamental generic logical structure to describe an enterprise through a two-dimensional descriptive schema. The schema is based on historical classifications of a six by six matrix with the communication interrogatives as columns and the reification transformations as rows [124]. The columns enable comprehensive, composite description of complex ideas through answering the question primitives: What, How, Where, Who, When and Why. In the context of software systems [125], the what column focuses on answering what data a system is made of, referring to system structure. The how column focuses on answering how the system works, referring to system functions. The where column focuses on answering where the system elements are located, referring to system architecture. The who column focuses on answering who performs the system functions, referring to system agents. The when column focuses on answering when things happen, referring to system behaviour. Finally, the why column focuses on answering why the system is needed, referring to the system's goals.
The rows represent different principal perspectives of the process of engineering including: planner, owner, designer, builder, implementer and user. Row 1, the planner, defines the scope of the enterprise in the context of its external environment. Row 2, the owner, defines the conceptual models of business requirements through capturing how management intends to use the business. Row 3, the designer, defines the models of the systems that represent the logical specifications for business implementations. Row 4, the builder, defines the system design for business and technology to realize the business. Row 5, the implementer, describes systems design for the products that will be used for implementation. Row 6, the user, describes the runtime of system implementation in alignment with the business [126].

The interaction between the rows and columns, cells, offers 36 distinct representations relevant to an enterprise [124]. An enterprise is a complex system as it constitutes many details and relationships, and such complexity can not be captured in a single architecture [127]. Zachman claims that the two dimensions are both primitive and comprehensive, and that the resulting cells constitute a complete description of an enterprise, although this claim is not proved mathematically. It has been observed that the six interrogatives are universal in conceptual modelling under the guise of different analogies or synonyms, and therefore constitute a fundamental modelling concept even if they may not form a complete set [120, 128].

The logical structure of the Zachman matrix offers a clear focus on a single variable in each cell, and at the same time maintains the integrity of the holistic view of the enterprise. This balance of holistic and pragmatic views simplifies understanding and communication of the enterprise architecture [118]. The framework does not replace the modelling processes but represents a system from many different perspectives and shows how they are related [129]. The framework does not offer an implementation or methodology [130], but on the one hand, it offers a flexibility to employ a wide range of modelling methods and notations. On the other hand, such a generic framework is difficult to apply in practice without a methodology for the implementation. This balanced between holistic and pragmatic views simplifies understanding and communication of the enterprise architecture [130], and there is a wide range of proposed research methodologies [131-134], tools [131-134], and recommended sets of modelling notations [125, 129, 132-136] for utilizing Zachman’s framework.
The structure of the QRMF proposed in this research is influenced by the structure of the Zachman framework, but focuses on the first three rows which represent the core concerns which RE addresses. Figure 16 shows the related software requirements view from these perspectives. A summary of concept mapping between RE views and the Zachman interrogatives is presented in Table 3.

<table>
<thead>
<tr>
<th>Scope Context (Executive)</th>
<th>Data (What (Things))</th>
<th>Function (How (Process))</th>
<th>Network (Location)</th>
<th>People (Who (People))</th>
<th>Time (When (Time))</th>
<th>Motivation (Why (Motivation))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inventory Identification</td>
<td>List: Inventory Types</td>
<td>Process Identification</td>
<td>List: Process Types</td>
<td>Distribution Identification</td>
<td>List: Distribution Types</td>
<td>Responsibility Identification</td>
</tr>
<tr>
<td>Business Concepts (Owner)</td>
<td>Inventory Definition</td>
<td>Process Definition</td>
<td>Business transform &amp; input/output</td>
<td>Distribution Definition</td>
<td>Business locations &amp; connectors</td>
<td>Responsibility Definition</td>
</tr>
<tr>
<td>System Logic (Designer)</td>
<td>Inventory Representation</td>
<td>Process Representation</td>
<td>System transform &amp; input/output</td>
<td>Distribution Representation</td>
<td>System locations &amp; connectors</td>
<td>Responsibility Representation</td>
</tr>
</tbody>
</table>

Table 3 Mapping Requirement Engineering Views into Zachman Framework

- What (data): this perspective addresses the informational entities (data, resources) produced or manipulated by the system.
- How (function): this perspective addresses the operational components performed by the system.
- Where (network): this perspective addresses the organizational architecture in which the software functions and objects are located.
- Who (people): this perspective addresses the agents responsible for performing system functions.
- When (time): this perspective addresses the behavioural perspective of software operations within the environment.
- Why (motivation): this perspective addresses the goal to be realized through the system.

Figure 16 Zachman Framework: Executive, Owner, Designer Perspective [137]
4.3 Curtis’s Modelling Perspectives for Software Process

Curtis [119] proposed the need to integrate multiple presentational paradigms to produce a complete software process model. This integrated paradigm should address what is going to be done, who is going to do it, when and where it will be done, how and why it will be done, and who is dependent on its being done [119]. The following four modelling perspectives are commonly used to answer these questions:

- **Functional perspective**: represents what are the processes to perform, and what are the flows of information entities relevant to these processes.
- **Behavioural perspective**: represents when to perform the process elements, i.e. the flow of the process elements which are subject to sequencing, feedback loops, iteration, complex decision-making conditions, input/output criteria.
- **Organizational perspective**: represents where and by whom the process elements are performed. This includes agents, communication medium and their distribution.
- **Informational perspective**: represents information entities produced or manipulated by a process, and relationships among them including data, artefacts, products and objects.

The process models can be used to facilitate human understanding, to automate process descriptions, to set a standard for actual process execution in the organization, to provide a framework for analyzing processes, and to automate actual processes [119]. To accommodate these goals, a model must be capable of providing its users with different types of information, such as what are the activities of a process (functional perspective); what information is manipulated in those functions (informational perspective); when and how the functions are performed (behavioural perspective); and where and by whom the functions are performed and data manipulated (organizational perspective). A single model can not represent all these perspectives without overcrowding the model and making it difficult to read and understand. Therefore, different Curtis models are used to model different facets of a system [119].

This approach does not introduce a new modelling paradigm to capture the integrated view of software processes nor coordinate the components of different modelling paradigms. Instead, Curtis proposed a modelling strategy based on a loose integration of various modelling paradigms [119]. A common denominator schema is defined containing only the core process information handled by various representations.
at high abstraction levels, the meta-model. This approach of modelling support is also adopted in this research, where the core concepts of different requirements views captured at a meta-level to guide adopting the QRMF without limiting it to specific modelling languages.

4.4 Integrative Approach to Multi-Perspectives Business Process Modelling

An integrative approach to support multi-perspective Business Process Modelling (BPM) was introduced to create comprehensive process models and to facilitate a common understanding of business perspectives regardless of the languages that represent them [120]. This approach is supported by two major processes. The first classifies business domain knowledge according to the six facets of Zachman’s Framework (What, How, Where, Who, When, Why). These facets represent the concepts of business domain knowledge including: Informational, Functional, Agent, Behavioural, Location and Goal. Table 4 summarizes the mapping of the business process concept into Zachman models.

Table 4 Mapping Business Process Schema onto Zachman Models [120]

<table>
<thead>
<tr>
<th>Zachman Model</th>
<th>Business process Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>What (Data)</td>
<td>Recourse</td>
</tr>
<tr>
<td>How (Function)</td>
<td>Process/Activity</td>
</tr>
<tr>
<td>Where (Network)</td>
<td>Containers</td>
</tr>
<tr>
<td>Who (People)</td>
<td>Agent</td>
</tr>
<tr>
<td>When (Time)</td>
<td>Event</td>
</tr>
<tr>
<td>Why (Motivation)</td>
<td>Goal</td>
</tr>
</tbody>
</table>

The second process selects a combination of models from modelling languages to represent perspectives. The selection is based on how close the language’s properties are to the business concepts identified from the requirements specification. This selection is supported by a modelling language ontology that contains the properties of modelling languages under each concept of the business domain knowledge. The matrix shown in Figure 17 illustrate the selection process: analysts start from the modelling language of their choice then step through the matrix to choose the language that complements their initial choice.
This approach provides a comprehensive business process model that incorporates the modelling perspectives of business domain knowledge. The main benefit of this approach is to provide a platform for integrating a selection of existing modelling languages where business analysts can use their favourite or familiar notations with notations that complement their choice, instead of introducing a new modelling language.

4.5 Relevance of Multi-Perspectives Modelling Framework to QR Modelling

The multi-perspective modelling approach has been adopted to address the various interrelated concepts found in different research areas, such as enterprise architecture [118, 122, 138, 139], information system [24], [104] and process modelling [119, 120]. The common objectives of deploying a multi-perspectives modelling paradigm can be grouped into two categories. The first category comprises the products/outputs of the modelling paradigms, and the second is the efficiency of the modelling process. A summary of these objectives follows.

Modelling Product

- To construct a comprehensive and complete model composed of diverse concepts related to the universe of discourse.
- To avoid scattered or overcrowded models difficult to understand, use and modify.
• To allow in-depth analysis, representation and manipulation of a specific set of concepts.
• To facilitate management of change where changing one model does not affect the others and it is also easy to track the changes.
• To bridge the gap between information systems concepts and business/enterprise concepts.

**Modelling Process**

• To improve the processes of software development, enterprise architecture, process reengineering, etc. where well defined models and documentation can be a foundation for long-term productivity and quality.
• To understand, compare and evaluate modelling languages and techniques.
• To offer a comparison framework for selecting suitable modelling techniques of individual project without imposing specific modelling language.

Table 5 lists the modelling frameworks supporting each of these objectives.

<table>
<thead>
<tr>
<th>Multi-perspective modelling objectives</th>
<th>Modelling Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construct comprehensive models</td>
<td>Zachman [118], 4+1 View model [24], Integrated Approach for BPM [120], Curtis’ process modelling [119], MEMO [122], Sommerville [82]</td>
</tr>
<tr>
<td>Avoid scattered or overcrowded models</td>
<td>Zachman [118], 4+1 View model [24], Integrated Approach for BPM [120], Curtis’ process modelling [119], MEMO [122], STATEMATE- based modelling [140], BP oriented RE process [121]</td>
</tr>
<tr>
<td>In-depth manipulation of concepts</td>
<td>Zachman [118], 4+1 View model [24], Integrated Approach for BPM [120]</td>
</tr>
<tr>
<td>Management of change</td>
<td>Zachman [118, 141], Curtis’ process modelling [119], STATEMATE- based modelling [140], ARIS [138], CIMOSA [139]</td>
</tr>
<tr>
<td>Bridge the gap between information systems concepts and business concepts</td>
<td>Zachman [118], Integrated Approach for BPM [120], MEMO [122], ARIS [138], CIMOSA [139], BP oriented RE process [121], Taxonomy of BP [142]</td>
</tr>
</tbody>
</table>

**Modelling Process**

<table>
<thead>
<tr>
<th>Modelling Process</th>
<th>Modelling Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process improvement</td>
<td>Zachman [118], Curtis’ process modelling [119], CIMOSA [139], BP oriented RE process [121]</td>
</tr>
<tr>
<td>Evaluate modelling techniques</td>
<td>Integrated Approach for BPM [120], Curtis’ process modelling [119], Taxonomy of BP [142]</td>
</tr>
<tr>
<td>Select modelling techniques</td>
<td>Integrated Approach for BPM [120], Taxonomy of BP [142]</td>
</tr>
</tbody>
</table>
Letsholo, Chioasca and Zhao [120] observed that the Zachman framework constitutes the most comprehensive and generic multi-perspective modelling, where the questions composing Zachman’s columns are the universal concepts of modelling stated under the guise of different synonyms: What (entity, object, subject, material, resource, information, thing); How (operation, process, function, activity, task, action); Where (location, place, container, network, structure); Who (agent, actor, people, machine); When (time, event, exceptions); Why (motivation, purpose, goal, objective, strategy, policy, desired state) [120].

### Table 6 Mapping Zachman’s Rows to Software Development Phases

<table>
<thead>
<tr>
<th>Zachman Rows</th>
<th>Software Life cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope and context (Planner)</td>
<td>Requirement specification, strategic planning</td>
</tr>
<tr>
<td>Business Model (Owner)</td>
<td>Requirement specification, business modelling</td>
</tr>
<tr>
<td>System Model (Designer)</td>
<td>Requirement specification(system modelling)</td>
</tr>
<tr>
<td>System operation (Customer/user)</td>
<td>System design</td>
</tr>
<tr>
<td>Technology Model (Builder)</td>
<td>System design</td>
</tr>
<tr>
<td>Detail Implementation (Programmer)</td>
<td>Implementation</td>
</tr>
</tbody>
</table>

### Table 7 Mapping Zachman’s Rows to Requirement Process

<table>
<thead>
<tr>
<th>Zachman Rows</th>
<th>Requirement focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope and context (Planner)</td>
<td>Strategic context to capture business requirements</td>
</tr>
<tr>
<td>Business Model (Owner)</td>
<td>Tactical model to capture detailed business requirements</td>
</tr>
<tr>
<td>System Model (Designer)</td>
<td>Technical model to capture system requirements</td>
</tr>
</tbody>
</table>

On the other hand, the perspectives represented in Zachman’s rows are the roles of the design process. This reflects the gradual development of modelling knowledge starting from the most abstract strategic level down to the details of models’ enactment [124]. Letsholo, Chioasca and Zhao [120] observed that the Zachman framework constitutes the most comprehensive and generic multi-perspective modelling, where the questions composing Zachman’s columns are the universal concepts of modelling stated under the guise of different synonyms: What (entity, object, subject, material, resource, information, thing); How (operation, process, function, activity, task, action); Where (location, place, container, network, structure); Who (agent, actor, people, machine);
When (time, event, exceptions); Why (motivation, purpose, goal, objective, strategy, policy, desired state) [120].

Table 6 shows the mapping of Zachman’s framework rows to the five basic phases of the software development life cycle. Capturing the knowledge related to each perspective effectively improves the quality of the modelling process [119, 121, 122, 139] and also bridges the gap of knowledge between different perspectives [121, 122, 138]. In the requirements process, the focus of this research, the requirements elements of business and software systems are covered in the first three perspectives of Zachman’s framework [121], as shown in

Table 7.

Observing the features of different multi-perspective modelling approaches, we conclude that a complete and comprehensive requirements model can be compiled through the six modelling views of Zachman’s columns and the first three perspectives of Zachman’s rows. The six views represent requirements views (goal, agent, function, structure, architecture and behaviour) and the three perspectives represent requirements development phases of different business levels (strategic, tactical and technical). Therefore, this research will model QR through these 18 perspectives to overcome the limitations stated in Chapter 3, and to achieve the same common objectives of multi-perspectives modelling summarised in Table 5.

4.6 Summary

Chapter 3 suggested that the multi-perspectives modelling approach can address the limitations of both QR classification models, introduced in Chapter 2, and requirements modelling frameworks, introduced in Chapter 3. This chapter therefore aimed to identify the features of multi-perspectives modelling frameworks that can be adopted for the development of QRMF. The most comprehensive framework is Zachman’s, as it comprises all the concepts found in multi-perspectives modelling frameworks. The first three rows are relevant to the requirements process and are therefore adopted in the QRMF design. The following chapter details the design of QRMF.
5 QRMF: Quality Requirements Modelling Framework

5.1 Introduction

This chapter describes the multi-perspectives framework for QR modelling that has been developed throughout the course of this research. The framework offers a process-oriented approach to modelling QR, focusing on how to capture and express QR from different views and phases of requirements. The main aim of this research is to provide a framework that helps requirement engineers in the process of constructing a fairly complete and comprehensive model of the QR of a given project. In order to do this, the entire RE modelling views and the related QR are brought together in a descriptive representation schema that cohesively includes all of them. The schema represents a logical structure that classifies and organizes the representations of QR of different requirement concerns and modelling views. This structural representation plays a significant role in guiding the construction of requirements models comprehensively and with consistency.

This chapter expresses the QRMF through detailing the framework components and the rationale behind its design. Section 5.2 gives an overview of the QRMF components, which are detailed in the remaining part of this chapter and the following two chapters. Sections 5.2 and 5.3 detail the QR concepts addressed and how to express them in the framework. Section 5.4 describes the framework's representational schema structure. Section 5.5 details the modelling perspective with the related QR concepts of each cell in the representational schema. Section 5.6 highlights the framework's features and contributions. Finally, section 5.7 summarizes the chapter.

5.2 QRMF Overview

The QRMF enables the construction of fairly complete and comprehensive QR models through addressing the limitations of current QR modelling approaches. The framework is based on a multi-perspectives modelling approach with the support of a structural schema and a modelling process. It has five components, three modelling and two abstract, as illustrated in Figure 18. The modelling components are the means to integrate QR concepts into FR models for specific software systems. The abstract components are blueprints of the framework's concepts and relationships that can be adopted in the development of modelling tools or ontologies to support the framework.
The QR Description Form includes concepts to define the QR of a software product in a traceable and unambiguous fashion through expressing the concepts related to identify, analyze and validate the QR. This form can be used by requirements engineers or business experts when describing the QR related to the software system. More details of the QR description form can be found in section 5.3.2. A blueprint of the concepts and relationships among those concepts is captured in the QR Description Schema. The schema is described using an entity relationship model to be used in designing tools to support the framework. More details related to the QR description schema can be found in section 5.3.3.

The framework’s Representation Schema structures requirement models into a two-dimensional matrix. This structure helps in acquiring the necessary knowledge systematically through classifying and organizing the descriptive representations of QR from different views and at different levels of requirements modelling. Both dimensions represent generic requirement concepts to facilitate utilizing the framework using various RE approaches and modelling notations. The rows represent strategic, tactical and operational levels which reflect the flow of the requirement development phases, from early to late requirements, and characterize the different levels of stakeholders’ knowledge. The columns represent goal, agent responsibility, structural, functional, architectural and behavioural viewpoints which cover all RE modelling views. The intersection between a row and column forms a unique perspective of the requirements model composed of specific functional and QR concepts. Conversely, when the cells are grouped together they form a complete requirements evolution model for the development of requirements integrated with the related QR concepts. More details related to the framework representation schema can be found in sections 5.4 and 5.5.
The QRMF has four levels of abstraction: level 3 is the most abstract representing the meta-meta-model and level 0 is the most detailed representing runtime scenarios. Level 0 is outside the scope of this research as it related to model execution. Level 1 refers to the models constructed for a specific application which is guided by the framework modelling process detailed in Chapter 7. Level 2 refers to the framework meta-model which is described as the framework representational schema. Level 3 refers to the meta-model of the framework representational schema; this meta-model is made up of six core concepts that link requirements modelling views with the related QR concepts. Such a high-level of abstraction offers a blueprint which can be used to develop a modelling tool to support the framework without restricting the modelling notations adopted in the framework. More details related to the meta-models can be found in Chapter 6.

The QRMF modelling process offers a constructive method for building QR models using the framework. The process consists of three phases, shown in the rows of the framework’s representational schema. At each phase a method elaborates the process of identifying and expressing the concepts related to each cell on the row. The process is generic and can be adopted without restricting the requirements modelling approach or modelling notation. More details related to the framework modelling process can be found in Chapter 7.

5.3 Quality Requirements Concepts

Before elaborating on the framework structure and how the QR are modelled, this section describes the QR concepts addressed in order to understand the framework approach to modelling QR and to justify its contributions. The section starts with a brief introduction about different QR concerns, then defines and justifies the concerns addressed in the framework. The concerns related to software product QR are addressed as they are relevant to the core set of functional requirements; section 5.3.1 details these concerns. Section 5.3.2 introduces the QR description form that depicts the components related to QR identification, analysis and validation. This helps to express QR in a way that they can later be traced and measured. Finally, section 5.3.3 introduces the QR description schema that depicts the syntax for connecting the identified QR semantics. The schema is designed as an Relational schemato depict a simplistic view of concepts and interrelations which can be easily adopted by database management systems and XML documents.
5.3.1 Quality Requirements Concerns

The great diversity of QR concerns have been classified into interface, operation, performance, life cycle, political and economic concerns [49]. Addressing QR without referring to the related concerns can lead to a conflict within the model. For example, the availability in operation concern means the time taken to access the service, or hours of operation; whereas, the availability in life cycle concern means the ability to access and update development resources, i.e. models or code. Although it has been possible to individually define the QR related to a specific concern [49], a complete model can not be constructed if we split them because they are interrelated. For example, the response time of performance concerns is affected by the response time of interface concerns.

To reach a trade-off between individual model concerns and to interrelate them, we grouped the concerns into two categories: QR related to the deliverable software product, which emphasize the interest of business audiences; and QR related to the software development process, which emphasize the interest of technical audiences. The QR related to the deliverable software product include interface, performance, operational, economic and political concerns. The interface concerns are related to the language of interactions, and its QR include understandability, learnability and attractiveness. The performance concerns are related to software runtime, such as response time, throughput, reliability, availability and security. The operational concerns are related to the environment conditions that constrain software operation and involve QR such as workload, peak hours, personnel availability, personnel skill level and hardware performance. The economic concerns are related to runtime costs such as services, licences and cost of resources. The political concerns are related to policies, standards and legal constraints that affect the software QR, such as compliance with the NHS access policy [143], encryption standards and security certificate. Table 8 summarizes the QR concerns related to the deliverable software product.

On the other hand, the QR related to the development process include operational, life cycle and economic concerns. The QR of operational concerns are related to the ability of the software product to be modified for improvement and its adaptability to change. This concern includes QR such as maintainability, stability, flexibility and changeability. The QR life cycle concerns are related to software development constraints and the development process, such as development time, resource availability, complexity and methodology. The QR economic concerns are related to immediate cost (development and setup) and long-term cost (updates, training and maintenance). Table 9 summarizes the
QR concerns related to the development process. Although the QR concerns related to the software development process impact the life span of the software product and may depend on software product requirements, they do not directly relate to the software core requirements. It is also possible to model them independently as they address different concerns and target a different audience. The framework focuses on modelling the software product QR concerns which are related to the core set of software requirements.

**Table 8 QR Concerns Related to a Software Product**

<table>
<thead>
<tr>
<th>Concerns</th>
<th>Quality Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>Communications: responsiveness, accessibility, operability, compliance and other QR related to the performance of communications. Interface: understandability, learnability, attractiveness and other QR related to the performance of interfaces.</td>
</tr>
<tr>
<td>Performance</td>
<td>QR of the software at runtime such as response time, throughput, reliability, availability and security</td>
</tr>
<tr>
<td>Operational</td>
<td>Environment conditions including work load, peak hours, personal availability, personal skill level and hardware operational performance</td>
</tr>
<tr>
<td>Economic</td>
<td>Runtime cost including services, licences and resources cost.</td>
</tr>
<tr>
<td>Political</td>
<td>Policies, standards that constraints software QR</td>
</tr>
</tbody>
</table>

**Table 9 QR Concerns Related to the Development Process**

<table>
<thead>
<tr>
<th>Concerns</th>
<th>Quality Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operational</td>
<td>The QR are related to the ability to improve the software such as maintainability, extendibility, and changeability.</td>
</tr>
<tr>
<td>Life Cycle</td>
<td>The QR are related to software development constraints and the development process such as development time, resource availability, complexity and methodology.</td>
</tr>
<tr>
<td>Economic</td>
<td>The QR are related to immediate cost (development and setup) and long term cost (updates, training and maintenance)</td>
</tr>
</tbody>
</table>

**5.3.2 Quality Requirements Description**

This section introduces the QR description form which includes elements used to express the QR in an unambiguous and traceable fashion. The aim is to offer a rich treatment of QR concepts related to identifying, analyzing and validating the QR. QR identification captures and defines stakeholders’ QR within their context to avoid ambiguous definition. The QR analysis transforms the identified stakeholder QR into a technical view which can be measured and verified objectively. The QR validation ensures the validity and traceability of the technical QR view against the QR identified initially.
The identification of QR starts by capturing stakeholders' expectations, demands and needs [14, 19]. It is common in the early phase of the requirements process to express QR vaguely and subjectively, as stakeholders initially have an immature understanding of requirements. It is also true that not all the information captured in the early requirement phase will be used in the design and implementation of the system. Although the level of understanding develops throughout the requirements process, it is important to capture and trace from the beginning the elements related to the domain and stakeholder expectations in a clear descriptive fashion. The early requirements phase facilitates a deep understanding of the enterprise setting, and this information is used to generate the correct requirements [95]. Expressing these elements explicitly helps to clearly define QR within their context. These elements include the QR type, topic, stakeholder, stakeholder's expectation, propriety, description and rationale. Another important element introduced in this research is the QRID, a QR identification mechanism, that flags each QR with an ID that can be traced throughout the requirements development and also helps to trace whether the QR meets the owner's expectations.

Type is how the terminology expresses QR such as security, performance and accessibility [19]. These terminologies are normally classified in hierarchical catalogues [19, 21] of QR types/characteristics and sub-types/sub-characteristics. This is insufficient to express a meaningful QR unless the QR topic is identified by type [19], such as an account (data object), report generation (function) and advisor (user). The combination of type and topic performs the basic meaningful definition of a QR that forms the objective to be achieved [19], such as the security of an account, performance of report generation and accessibility to an advisor. The description can be optionally expressed for further explanation of the QR. It is also essential to express the QR's owner and their expected quality level; for example, an account holder would expect high account security and a physician would expect to receive a radiology report within four weeks. QR prioritization will distinguish criticality and domination QR [19]: the critical QR has a high degree of importance and the dominant QR influences different parts of the system. Prioritization is important to justify design decisions, especially in cases of conflict between a QR and limited resources. The priorities can be modelled through a numerical scale or qualitative phrases. In the former, a bigger number represents higher priority and a lower number represents lower priority. The qualitative phrases flag the QR with a very high, high, average, low or very low priority tag. This rationale justifies why a QR is required to help
understanding the business drivers behind it. Table 10 summarizes these elements in a QR description form for an early requirement phase with examples.

Table 10 QR Description Form for Early Requirement

<table>
<thead>
<tr>
<th>Description Elements</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>QRID</td>
<td>1, 1.1., 1.2., 1.1.1., 1.1.2.</td>
</tr>
<tr>
<td>Type</td>
<td>Security, Reliability</td>
</tr>
<tr>
<td>Topic</td>
<td>Account, Transaction</td>
</tr>
</tbody>
</table>
| Quality Requirement  | 1. Security of an account  
|                      | 2. Reliable transaction |
| Description          | 1. Secure an account from unwanted actions  
|                      | 2. Ensure correct transaction |
| Rationale            | 1. Accounts info is private and sensitive  
|                      | 2. Gain customer trust |
| Stakeholder          | 1. Bank manager  
|                      | 2. Account holder |
| Expectation          | 1. 99.99% account security  
|                      | 2. 99.99% reliable transaction |
| Priority             | 1. Very high  
|                      | 2. Very high |

The QR analysis looks at the QR from an engineering point of view. It transforms the identified business perspective of QR into an achievable and measurable technical perspective. The model expresses elements related to QR refinement, operationalization measurement and selection rationale. The refinement decomposes the QR into a more specific and simpler form, helping to clarify the QR which may initially have been stated briefly and ambiguously [19]. For example, account security is refined to confidentiality of the account and integrity of the account. The refined QR is a specialized form of the identified objectives and is therefore expressed as described in the previous paragraph. The refined QRID follow a sub-level sequence of the parent QR; for example, if the parent QRID account security is 1 then the confidentiality of the account QRID is 1.1 and the integrity of the account QRID is 1.2. A QR may be refined through several levels until it can be operationalized. At the lower level of refinement, the QR matrix is identified with its related measures, just prior to operationalization, as it is the simplest form of expressing the QR. The matrix is a method that estimates possible values to apply in measuring the achievement of the QR. The measures are matrix variables; they normally refer to the
software and environment operational attributes. Examples of software attributes that measure confidentiality are the total numbers of account access, numbers of fraudulent access and numbers of confidential access. Examples of environment attributes are a maximum number of current transactions. The matrix for this calculates the confidentiality of the account = $1 - \frac{\text{Average of fraudulent access}}{\text{Total number of access}}$; this value is used to evaluate the fulfilment of the stakeholder’s expectation, which in this case is 99.99% confidential access. The real values of attributes are related to the software system operation; at this phase requirement engineers only specify the matrix and related attributes which are going to be used at software testing and to trace software quality during operation. However, not all QR can be measured through such a quantified approach, for example friendly interface. A qualitative form of measurement can be applied, such as interview or observation. The framework does not restrict specification of one form or the other, but offers guidance in identifying all possible QR factors.

The operationalization identifies the possible options for achieving the refined QR [19]; for example, the confidentiality of an account can be achieved by access control such as UserName/password, biometrics, or encryption, while the integrity of the account can be achieved by validating account access against eligible rules. The operationalization ID follows the same sequence for refining the QRID; for example, the three options to realize confidentiality with QRID 1.1: User Name/password QRID is 1.1.1, biometrics QRID is 1.1.2, and encryption QRID is 1.1.3. The model also expresses the influence of operationalization alternatives to support or hinder the QR objectives, which consequently help selection decisions among these alternatives. As already stated, another important factor that also influences the selection decision is prioritization among related QR objectives. For example, applying encryption to secure the account information can slow access to the account; however, security has higher priority than swift accesses. Table 11 extends the QR description form with elements related to the later requirements phase, with examples.

Requirement validation is conducted at the later requirement phase where only requirements relevant to the software are identified and modelled. The QR validation verifies that the technical QR perspective complies with the initially identified stakeholders’ QR. The framework facilitates tracing the QR through following the sequence of QRIDs, so the requirements engineer can review the QR evolution in forward or backward fashion. Tracing the QR helps checking the unambiguous, consistent, conformant and completeness attributes of QR models. The unambiguous check ensures each QR has only one interpretation. The consistency check ensures that no QR conflicts
with another. The conformance check ensures that each QR correctly meets user expectation. The framework facilitates checking these three validation characteristics through its ability to trace a QR and its sub-levels via the sequence of QRIDs which refer to the QR definition within a specific context. Finally, the completeness check ensures that all stakeholder requirements are refined until operation; when the operation is missing this implies that the QR is not properly defined or analyzed sufficiently. Validated QR need to be ticked to avoid redundancy and, that no additional semantics are needed for QR validation.

**Table 11 QR Description Form for Late Requirement**

<table>
<thead>
<tr>
<th>Description Elements</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>QRID</td>
<td>The refined QR for account security with 1. QRID will take 1.1, 1.2...etc.</td>
</tr>
</tbody>
</table>
| Refined QR           | 1.1. Confidentiality of the account  
                        1.2. Integrity of the account  
                        The examples of 1.1. Confidentiality of the account and 1.2. Integrity of the account can be expressed with the same semantics shown in Table 10. |
| Matrix               | 1.1. Confidentiality of the account = 1 - (Average of fraudulent access) |
| Expectation          | 1.1. 99.99% confidentiality of an account |
| Attributes           | Attributes to measures 1.1. Confidentiality of the account includes:  
                        software attributes:  
                        • Total number of account access  
                        • Number of fraudulent access  
                        • Number of confidential access  
                        Environment attributes:  
                        • Maximum number of current transaction |
| Operationalization   | 1.1.1. Authorization with UN/PW  
                        1.1.2. Biometrics  
                        1.1.3. Encryption  
                        1.1.4. Access reader  
                        1.2.1 Validating access against eligible rules |
| Refinement link       | 1.1.3 Encryption is used with other operationalization options of 1.1 account confidentiality, therefore it is linked with other options through And relationships. OR relationship is used to show a selection among alternative operations (1.1.3. AND 1.1.1.) OR (1.1.3. AND 1.1.2.) OR (1.1.3. AND 1.1.4.) |
| Contribution link     | An example of how a QR can influence another QR is 1.1.3. Encryption hurt 3.1. transition speed where the QR priorities are used for selection |
5.3.3 Quality Requirement Description Schema

The previous section introduced the QR description form of elements related to QR identification, analysis and validation in the early and later phases of the requirements process. These elements are various and interrelated, and can be difficult to grasp and trace. This section proposes a QR structural schema that depicts the syntax linking the above QR elements and the relationships among them. Figure 19 illustrates the QR elements using an relational schema. The schema proposed depicts the grammatical structure that links the QR elements after normalization, aiming to minimize redundancy and offer a flexible query structure of QR elements. Redundancy is eliminated by separately defining the repeated elements then linking them to the QR definition. The repeated elements mainly reflect attributes and objects that characterize the definition of a QR, such as priority, stakeholder and account information. Once the elements are separated in the structural schema there is flexibility to query and group QR. The Relational schema can easily be adopted by a database management system or XML documents, which can be used to develop a tool for modelling, analysis and validating QR. Although this research does not propose a tool, a grammatical structure that links the QR elements can be easily defined. The following is a brief explanation of the schema introduced in Figure 19.

![Figure 19 QR Description Schema](image)

The quality types, topics and owners are placed as separate objects, which facilitates grouping QR by each of these elements; for example, search for all the QR
demanded by a bank manager, search for all the QR related to an account, or join these queries. The QR object is the main object in the schema and is linked to most of the other objects and the QR's attributes. The priority object defines the prioritizing mechanism among QR, which influences decision making and trade-off among them. The parent ID is to flag the refined form of the parent QR, which helps to trace different levels of refinement. The QR refinement object depicts the relationship between the refined QR through AND/OR refinement types. The matrix and attribute objects help to define the matrix for measuring the QR through defining related formulae, attributes, values and units. The operation object depicts all the possible options to realize the QR. The relationship between the refined QR and possible operations is depicted by the operation refinement object, which identifies AND/OR refinement types. Finally, the contribution relationship object depicts the contribution type between operation and the QR, which influences decision making related to the QR and its operations.

5.4 QRMF Representational Schema Structure

As was illustrated above, the framework attempts to address the complexity of QR through integrating QR with different FR models. This section explains the rationale behind using different requirements models and including QR as a part of those models. This is followed by explaining how the framework places these models in a structural schema that classifies and organizes the FR concepts with the related QR elements. The representational schema is the core of the framework as it offers a flexible guide for expressing the entire requirements in a systematic fashion. The details of the modelling perspectives composing the framework's representational schema are expressed along with an illustration of the FRs and QR modelling elements in a two-dimensional matrix.

Requirements modelling is an iterative process that evolves as stakeholders come to understand the business domain and actual requirements [68, 73, 80]. Throughout the iterative process different levels of detail are revealed to extend this understanding. These levels evolve from abstract descriptions of business requirements in the early phase of the requirements process to detailed technical requirements relevant to the system design in the later phase. At each level of detail, various types of model are used to express requirements, each of them considering specific views of the requirements. Given this diversity, a single model can not completely express all the requirements [24]. Thus, a comprehensive expression of requirements is conducted through a multi-perspective modelling approach, such as the multi-view modelling framework [80], 4+1 view model
[24], KAOS [144] and adaptation of the Zachman framework [124] for requirements modelling [145]. Although a wide range of multi-perspective modelling approaches is described in the RE literature, none of the approaches integrates modelling QR with FRs models, as they are more complex and easier to miss than FRs; therefore, this framework addresses this gap.

Although some of these approaches are widely used, such as KAOS [144] and the 4+1 view model [24], the framework does not extend any of these approaches to express QR. Instead, it classifies the requirements modelling perspectives at abstract levels to form a generic structural schema. The generic schema gives the framework the flexibility to be adopted using different requirements approaches, design methods, notations and tools. The schema represents a logical structure that organizes modelling perspectives into a two-dimensional matrix. The matrix summarizes requirement development levels and modelling views related to each of these levels. The structure plays a significant role in guiding the construction of requirements models as it is designed with consideration of the subsequent stages of the requirements development process. Although the classification of requirements concepts is different from the enterprise architecture concepts, the framework was strongly influenced by two features of Zachman's framework [124]: first, the generic schema of the Zachman framework that made it widely adopted; and secondly, the matrix of rows and columns which provides a logical structure of concepts and guides the development of requirements models.

The structure of the QRMF schema consists of three rows and six columns. The rows represent the transformation of requirements from the business to the technical level. This transformation evolves throughout requirements development, and consists of several common activities which can be summarized as elicitation, identification, modelling, analysis, validation and specification. The output of these activities can be grouped into three generic levels: strategic, tactical and operational. The columns represent the universal communication questions to describe complex ideas: Why, Who, What, How, Where and When. In RE each of the communication interrogatives expresses a view of the requirement model: Why expresses the goal view, who the agent responsibility view, what the structural view, how the functional view, where the architectural view and when the behavioural view. These six modelling views encompass the entire RE modelling views [B0]. These generic perspectives of the framework schema are illustrated in Figure 20. The following two sections present a detailed description of these perspectives, and the framework rows and columns, including the related QR elements.
5.4.1 Row One: Strategic Level

The strategic level conveys a high level of detail related to a business or organization. It concerns coarser-grained goals of the organization, the stakeholder who demands these goals, the essential data to be maintained by the organization, the key functions or services to deliver, an outline of the architecture of the organization, and significant scenarios to consider or to avoid. These concerns convey business details that help to scope the project, identifying business objectives and related policies. These details are captured early in the RE process, where top-level strategic stakeholders are involved, such as top manager, CEO and CIO. In this context, all their needs and expectations are relevant even if they are conflicting or over-ambitious; resolution is made at later phases. The strategic details are typically captured through requirements elicitation and domain understanding activities. These details are normally expressed in textual form, such as preliminary draft proposals, tender documents, minutes of initial meetings, policy documents, etc. as this form is the basis of business communication.

The QR are also expressed in generic terms for the whole organization and specified at later phases of requirements. At this level the QR are stated as QR goals, legal constraints and economic constraints. Despite the generic nature of the strategic level, it is important to specify as much as possible from the elements stated in the early QR description form shown in Table 10. Specifying the type and topic are the minimum requisites to express a QR and assign a QRID. However, most of the other elements can be expressed at this high level, such as the stakeholder's demanded QR and his/her expectation.
5.4.2 Row Two: Tactical Level

The tactical level articulates the requirements obtained by requirements engineers and some business stakeholders at departmental level, after consolidation of the strategic level. It expresses requirements as goals, services and constraints that need to be recognized by the system. The tactical details are typically captured through modelling notation, which facilitates documenting and analyzing the elicited requirements. High-level architectural models are used to structure the tactical concerns which determine what the software will implement. The notations that can be used are the ones that offer an outline view of requirements, such as strategic dependency models and the strategic rational model of i*, to structure and refine the goals and related concepts of the intended system; and BPMN to illustrate scenarios and the architecture of the intended system.

The QR addressed at the tactical level is related to the refined goals, including performance, legal and economic constraints; and related to the overall system architecture including interface constraints. The QR becomes more specific as it is refined into finer-grained requirements at the level before the operational form. At this stage more elements of the QR description form for early and later requirements, shown in Table 10 and Table 11, can be expressed.

5.4.3 Row Three: Operational Level

The operational level is detailed in models for realizing requirements, which determine how the software will be implemented. It extends the primary structure of tactical models with related environmental conditions to include details about requirement realization. The operational level is developed throughout the requirements analysis and documentation and validation activities. These activities transform the business view of requirements into an operational view to realize them. Modelling notations facilitate capturing, expressing and verifying the interrelated details of the operational level. The Softgoal Interdependency Graph (SIG) of the NFR framework, functional model and behavioural mode are examples of the modelling notations that can be used. While requirement engineers are developing these models, some stakeholders at the operational level may also become involved.

Operational details of QR may require some refinement prior to reaching operationalized items. The refinement decomposes the QR into more specific forms, and operationalization identifies possible tactics to realize them. Sometimes a decision needs to be made to select among possible operationalizations or conflicting QR, and the NFR
Framework [19] offers the most comprehensive treatment for refinement, operationalization and selection of QR using SIG. In addition to the conceptual modelling of QR offered by SIG, QR can be included in other operational models that depict different views of the requirements; for example, integrating the QR related to the interface constraints with the architecture model, and integrating user skill levels and user performance constraints with the agent responsibility model. It is also important to identify QR measures, target values and expected environmental conditions, which can be used for validating the requirements and testing the software after development. At this stage the QR description form of late requirements, shown in Table 11, can be specified in addition to some elements from Table 10 which become clear in this level.

5.4.4 Column One: Goal View

The goal view describes the motivations and objectives of the system's stakeholders. This view answers the question of why the system is to be developed, including what objectives should be met, what problems should be solved and what risks should be avoided. Depending on the RE approach, goals can be stated in different forms such as stakeholder expectations, policies and standards or justification for a specific transaction. Many RE approaches consider the goal view as a core modelling paradigm for RE, such as KAOS [67], i* [14], an OMG standard BMM [146], etc. Most goal-oriented modelling approaches consider QR as soft-goals to be achieved which can also be refined, operationalized and validated.

5.4.5 Column Two: Agent View

The agent view represents the responsibilities of all the agents interacting with the system, and their expectations. This view answers the question of who are the agents interested in and interacting with the system. Agents are the stakeholders who have a legitimate interest in the system, whether internal or external, including persons such as end users, organizations and regulatory authorities. An agent model captures agents' capabilities, responsibilities, interfaces and hierarchical dependencies. Hierarchical dependencies group agents and their responsibilities into several levels which can influence the architectural view of requirements. The QR addressed in this view are related to the expected performance level (e.g. an account holder expects reliable account transactions); agent performance (e.g. constraints on customer service, such as must respond to enquiries within 24 hours); and agent skill levels required for system operation (e.g. minimum level of IT skills required).
5.4.6 Column Three: Structure View

The structure view of requirements modelling gathers a system’s conceptual objects, domain-specific concepts and the interrelations between them into an object model. This view answers the question of what objects compose the system. These objects can be described by a glossary of terms which can be used throughout the entire system development. The objects also resemble the information maintained in the system. The structural view can be modelled through an Relational schema or UML class diagram. In the RE process, an object model provides the concept definition and domain properties used by the goal, agent, operation and behavioural models [80]. The QR addressed in this view are related to performance constraints in accessing or retrieving an object such as secure account details; performance measures to construct an object instance, such as reliability of report generation; and performance constraints controlled by objects such as login object control and the time to access customer service.

5.4.7 Column Four: Function View

The function view of requirements modelling expresses the operations to be performed by the system in order to satisfy the goals. It answers the question of how to achieve the stated requirement goals. This model depicts the software's functional components with the related environmental conditions including inputs, outputs, and the pre- and post-conditions of an operation. Some types of functional model specify a dynamic view which includes data flow representing the input and output of functional components; and control flow representing connections between functional components. The operational model and UML use case diagram are examples of notations that can be used to model this view. The QR addressed are related to expected performance of a functional unit, such as the reliability of reports generated by ATMs; and performance constraints controlled or influenced by a function unit, for example a staff rota scheduling function to control the response time to customer enquiries.

5.4.8 Column Five: Architecture View

The architecture view expresses the physical distribution of system components including agents, sub-system components, storage and hardware. It answers the questions of where the system's components and functions are located. The flow among the system components and the interfaces between them is also illustrated. The architecture can be modelled through communication diagrams and business process modelling using BPMN. The QR addressed in this view are related to interface performance measures between components; for example, integrity of account information between service channels, ATM,
online, phone and customer service desk, or the waiting time to access customer service. Another form of architectural model is the organizational chart which depicts different levels of agents and their responsibilities. This model addresses the QR from the agent responsibility view with the same hierarchical relevance.

5.4.9 Column Six: Behaviour View

The behaviour view of RE expresses the permissible system behaviours resulting from external events and operations. This view of requirements modelling answers the questions of when a function or service takes place. Unlike the other modelling views, which offer an abstract description of the modelling concepts, the behavioural model offers a concrete description of scenarios which allows the explicit expression of system properties. Sequence models can capture the behaviour of a specific agent instance, and state transition models can capture a general class of behaviours. The QR addressed in this view are related to event trigger performance constraints, such as fund transfer time once an order is confirmed, and the QR related to scenarios such as increased security of account access after three authentication failures.

5.5 The Framework Cells – Representing Different Perspectives of QR

The previous section described the structure of the framework representational schema and the rationale behind the design of its rows and columns. The intersection of a row and column forms a cell with a unique perspective of requirements and QR. This section offers a detailed description of the matrix through illustrating the addressed requirements with the related QR in each of the cells. For example, cell (1,2) expresses the agent responsibility view from the strategic level, which depicts generic QR demands expressed as QR goals, legal constraints and economic constraints. Another example which depicts detailed operational specifications of QR can be found in cell (3,2) which expresses the agent responsibility view from the operational level. Conversely, when cells are grouped together, a fairly complete requirements model is formed reflecting the development of requirements with the related QR.

Figure 21 illustrates the requirements concepts with the related QR and modelling notations, expressing them for all the cells in the representational schema. A detailed description of each of the cells follows the figure, demonstrating the requirements concepts, the notation commonly used to express the requirement, and related QR with QR description elements and examples.
When attempting modelling in practice, the schema only restricts the order among rows as it follows the flow of requirements development. Therefore, the modelling practice must start from the first row, moving to the second and then the third. On the other hand, no specific order is recommended for the columns because these depend on the requirement modelling approach used. Furthermore, the relationships among the cells also vary according to the modelling approach. For example, in a goal-oriented modelling approach, refinement and operation relations guide construction of the models: the starting point is cell (1,1) as the first row illustrates a generic goal view of requirements with a generic illustration of softgoals; the second row is generated through goal refinement to illustrate more specific goals and softgoals, and the refinement continues until an operational goal is reached, as illustrated in the third row.

To take another example: when a process-oriented RE approach is used, the requirement practices may start from cell (1,4) or cell (1,5); to construct the models in the following rows, the decomposition relation can be used. Given this variation, the relations among the cells are not included in the definition of the schema. Furthermore, this gives the schema the flexibility to be used with a wide range of requirement modelling approaches. Nevertheless, the QR are still traceable throughout modelling practice through the QRIDs proposed in the QR description form.

5.5.1 Strategic Goal, Cell (1,1)

The strategic goal represents high-level objectives related to the whole organization. This cell captures why a system is needed through expressing the business objectives, policies and constraints associated with the organizational objectives. The notation used to express this information is a textual form, readable by business stakeholders with high-level decision-making authority. The QR addressed in this cell are related to the QR goals, legal constraints and economic constraints that must be considered while developing the software. The QR description elements that can be expressed in this cell are the QRID, type, topic, description and rationale. An example of a QR goal expressed in this cell is QRID 1, Type: Secure, Topic: Account information, Description: All financial records must be secure, Rationale: To build customer trust.
<table>
<thead>
<tr>
<th>Strategic goal (1,1)</th>
<th>Strategic agent (1,2)</th>
<th>Strategic structure (1,3)</th>
<th>Strategic function (1,4)</th>
<th>Strategic architecture (1,5)</th>
<th>Strategic behaviour (1,6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concepts: Organisation objectives answering why a system is needed</td>
<td>Concepts: Decision-makers answering who demand the objectives</td>
<td>Concepts: Key objects answering what are the things important to the business</td>
<td>Concepts: Function and services answering how to realise the business objectives</td>
<td>Concepts: Distributions of business components answering where the components are located</td>
<td>Concepts: Significant business scenarios answering when to trigger actions</td>
</tr>
<tr>
<td>Notation: Text</td>
<td>Notation: Text</td>
<td>Notation: Text</td>
<td>Notation: Text</td>
<td>Notation: Text</td>
<td>Notation: Text</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tactical goal (2,1)</th>
<th>Tactical Agent (2,2)</th>
<th>Tactical structure (2,3)</th>
<th>Tactical function (2,4)</th>
<th>Tactical architecture (2,5)</th>
<th>Tactical behaviour (2,6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concepts: Specific goals can be realised by the system</td>
<td>Concepts: Object related to the refined goals</td>
<td>Concepts: Functions related to the refined goals</td>
<td>Concepts: Distributions of business objects</td>
<td>Concepts: Business scenarios and the related environment variables</td>
<td></td>
</tr>
<tr>
<td>Notation: Expresses agent responsibility and demand link e.g. SD of 1&lt;sup&gt;1&lt;/sup&gt;</td>
<td>Notation: Expresses objects related to the refined goals and environment e.g. data object in BPMN and resource object SD of 1&lt;sup&gt;1&lt;/sup&gt;</td>
<td>Notation: Expresses functions interaction to agents, data and other function e.g. activity in BPMN, use cases and action object in SD of 1&lt;sup&gt;1&lt;/sup&gt;</td>
<td>Notation: Expresses location of system components, communication links, interfaces, access and information exchanged e.g. communication and architecture diagram</td>
<td>Notation: Expresses collaboration view that relates business events and timing sequencing rules to business actions, e.g., BPMN, flow chart and URL activity diagram</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operational goal (3,1)</th>
<th>Operational agent (3,2)</th>
<th>Operational structure (3,3)</th>
<th>Operational function (3,4)</th>
<th>Operational architecture (3,5)</th>
<th>Operational behaviour (3,6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concepts: Operational goals to can be implemented by system</td>
<td>Concepts: Agent responsibility model</td>
<td>Concepts: Software and environment entities captured by system</td>
<td>Concepts: Function related to system operation</td>
<td>Concepts: Distributions of system and communication among system objects</td>
<td>Concepts: Concrete description of system scenarios and related environment and operational properties</td>
</tr>
<tr>
<td>Notation: Tree structure for refinement and operationalisation e.g. SR of 1&lt;sup&gt;1&lt;/sup&gt; and SIG for NFR framework</td>
<td>Notation: Expresses dynamic of functional unit including input/output and environment condition</td>
<td>Notation: Captures system entities and their relationships e.g. class and ER diagram</td>
<td>Notation: Express location of system components, communication links, interfaces, access and information exchanged e.g. communication and architecture diagram</td>
<td>Notation: Expresses flow e.g. activity diagram and sequence diagram Object, transition e.g. state transition diagram</td>
<td>Notation: Expresses event that can trigger QRS QRS related to object of system scenarios</td>
</tr>
</tbody>
</table>

QRs Concepts: Soft-goals, legal constraints and economic constraints
QRs Concepts: Stakeholder’s QRs expectations and priority
QRs Concepts: QRs demand by an agent and QRS of agent performance
QRs Concepts: QRs of objects
QRs Concepts: QRs of objects
QRs Concepts: QRs of objects
QRs Concepts: QRs of objects
QRs Concepts: QRs of objects
QRs Concepts: QRs of objects
QRs Concepts: QRs to interface and protocol of interactions
QRs Concepts: QRs to interface and protocol of interactions
QRs Concepts: QRs to interface and protocol of interactions
QRs Concepts: QRs to interface and protocol of interactions
QRs Concepts: QRs to interface and protocol of interactions
QRs Concepts: QRs to interface and protocol of interactions
QRs Concepts: QRs related to the performance of functions or services
QRs Concepts: QRs of objects
QRs Concepts: QRs of objects
QRs Concepts: QRs of objects
QRs Concepts: QRs of objects
QRs Concepts: QRs of objects
QRs Concepts: QRs of objects
QRs Concepts: QRs of objects
QRs Concepts: QRs of objects
QRs Concepts: QRs of objects
QRs Concepts: QRs of objects
QRs Concepts: QRs of objects
QRs Concepts: QRs related to system’s functions

Figure 2.1: The QRMF Representational Schema
5.5.2 Strategic Agent, Cell (1,2)

The strategic agent represents the strategic decision makers of the overall organizational objectives. Those are high-level decision makers who determine the business objectives/strategic goals, which may include people or organizational bodies, internal or external to the organization. The notation used to express this information is a textual form, expressing the agents and their objectives. The strategic goal cell (1,1) and strategic agent cell (1,2) are closely related as they complement each other. The QR addressed in cell (1,2) are the agents’ expected QR goals and the priority assigned to achieve these expectations. The QR description elements are the stakeholder demands, the QR goals, their QR expectations and the priority level to meet those goals. An example of the QR description elements of the strategic agent related to the QR expressed in cell (1,1) is stakeholder demanding the QR goal with QRID 1: CIO, Expectation: 99.99% secure info and Priority: Very high.

5.5.3 Strategic Structure, Cell (1,3)

Strategic structure represents the main objects related to the strategic goals. This cell captures what the data or things related to business objectives. The notation is a textual form listing the objects included in the system. The QR addressed in this cell are related to the QR goals identified in cell (1,1) which is sometimes identified as a QR topic. For example, QRID 1 of the previous example is related to the account information object which is described as a topic of the QR goal.

5.5.4 Strategic Function, Cell (1,4)

Strategic function represents the main functional objectives to be achieved by the system. Those objectives refer to functions or services that specify how to achieve the business objectives. The notation is a textual form listing the functions and services performed by the software system. The QR addressed in this cell are related to the performance constraints of those services or functions. The QR description elements are the same as those specified in the previous cells; however, the topic refers to a functional objective. For example, QRID 2., Type: Reliable, Topic: Fund transfer, Description: all fund transactions must be reliable, Rationale: to build customer trust, Stakeholder: Operation manager, Expectation: 99.99% reliable transactions, and Priority: Very High.

5.5.5 Strategic Architecture, Cell (1,5)

Strategic architecture captures the distribution of strategic objects including data, agents and functions, i.e. the geographical locations of the strategic objects and the
organizational structure of agents. The notation is a textual form listing the geographical locations of the strategic objects or using the organizational chart to illustrate the agent hierarchy. The QR addressed in this cell are related to the interface and interaction protocols between strategic objects. The hierarchical levels among agents influence the decision making related to the QR. The QR description elements that can be expressed in this cell are the same as those specified in the previous cells; however, the topic refers to where to apply them. For example, QRID 3., Type: Availability, Topic: All payment points, Description: Customer can access money from anywhere at any time, Rationale: Customer convenience, Stakeholder: CEO, Expectation: 24hrs/7days Availability, and Priority: High.

5.5.6 Strategic Behaviour, Cell (1,6)

Strategic behaviour represents significant business scenarios to be achieved or avoided by detailing the interactions between organizational objects. This cell captures when a business action takes place and when problem scenarios may arise. The notation used to express this information is a textual form describing significant business scenarios. The QR addressed are the QR triggered by events or the QR that trigger an event. It also can be related to QR hazards that may result from an event. The QR description elements are the same elements specified for the previous cells; however, the topic refers to an event triggering a QR. For example, QRID 4., Type: Increase account security, Topic: Suspicious action is attempted, Description: Ensure account security when a suspicious action is attempted, Rationale: handle different security hazards, Stakeholder: Security manager, Expectation: 99.99% account security, and Priority: Very high.

5.5.7 Tactical Goal, Cell (2,1)

The tactical goal represents sub-goals or specific goals that collectively form a strategic goal. These sub-goals are captured through the goal refinement process, which identifies the sub-goals that compose strategic goals. The notation used to express different types and levels of refinement and the relationship among them is the tree structure. SD and SR models of i*, KAOS and the NFR framework for QR are examples of modelling notations that depict the tree structure of goals. The QR addressed in this cell are specified QR objectives, which are also captured through the goal refinement process. The QR description elements are the same elements that express the QR goals at the strategic level, in addition to the elements that describe the goal refinement relationships including refinement link and parent ID; and the elements that describe the specification of lower-level QR goals including the matrix and attributes. An example of how a QR goal is expressed in this cell is the refinement of the QR goal QRID 1 illustrated in cell (1,1) where
the refinement of secure account information includes QRID 1.1 confidentiality of account and QRID 1.2 integrity of account. The QRID 1.1, Type: confidentiality, Topic: account, parent ID: 1, Stakeholder: security manager, Expectation: 99.99% Confidentiality, Priority: very high, matrix: C=100(A/U) and attributes: A = No. access, U = No. unauthorized access.

5.5.8 Tactical Agent, Cell (2,2)

The tactical agent represents the demand and responsibility of agents in relation to different levels of goals. This cell captures which agent demand identified goals and who is responsible for undertaking them. Agent demands in terms of different levels of goals and QR goals are closely related to the goal model. Therefore, the notation used to express different levels of agent demand and responsibility is the tree structure expressed as the SD and SR models of i*. The UML use case and agent responsibility model of KAOS only capture operational goals. The QR addressed in this cell are related to the tactical agents, and include the demanded QR goals and the expected performance level of the agent who is responsible for realizing the operational goals. The QR description elements that express agent demand in this cell are the same as the elements included in the tactical goal cell (2,1). For example, the security manager demands secure access to an account. On the other hand, QR elements related to the agent’s responsibility are expressed as the QR of tactical functions in cell (2,4); for example, a credit agent responsible for reliable account access verification.

5.5.9 Tactical Structure, Cell (2,3)

Tactical structure represents the business objects of the refined goals and the domain specifications related to those goals. This cell captures what are the business objects of the refined goals and their domain specifications, which include data and resources. The notation used to express the tactical structure should be capable of detailing objects related to the refined goals and business environment, such as SD and SR of the i* framework, and BPMN notations. The QR addressed in this cell are the performance constraints for the identified business objects. The QR description elements that can be expressed in this cell are the same as the ones included in the tactical goal cell (2,1), where the topic refers to the tactical object; for example, secure access to account funding information.

5.5.10 Tactical Function, Cell (2,4)

The tactical function represents the functional units and related domain specifications needed to realize the tactical goals. This cell captures how to achieve the
refined goals, the responsible agents to perform these functions, and related environmental conditions. The notation used to express the tactical functions should be able to detail the functional components, responsible agent, the interrelation among functions and any relevant environmental specifications. The UML use case diagram and BPMN are examples of modelling notations that sufficiently represent the tactical function details. The QR addressed in this cell are related to functional performance measures. The QR description elements are the same as those included in the tactical goal cell (2,1), where the topic refers to the tactical function; for example, the system must secure fund transfer from user account.

5.5.11 Tactical Architecture, Cell (2,5)

Tactical architecture captures the distribution of business objects including data, agents, functional components and communication links among them. It captures the geographical locations of business objects and the communication links amongst them. The notation should be capable of detailing the tactical objects and their communication. The SD and SR models of the i* framework and BPMN are examples of modelling notations that depict the tactical architecture. The QR addressed in this cell are related to the interface performance of communication among business objects. The QR description elements are the same as those in the tactical goal cell (2,1); however, the topic refers to the tactical architecture; for example, minimize the waiting time in establishing communication between customer and customer service.

5.5.12 Tactical Behaviour, Cell (2,6)

Tactical behaviour captures the permissible behaviours of business scenarios and the related environment variables. The cell captures when to undertake business actions according to business events or timing sequencing rules. The notation used to express the tactical behaviour should be capable of capturing a collaborative view that relates business events and timing sequencing rules to business actions. BPMN, flow charts and UML activity diagrams are examples of notation that can be used to model tactical behaviour. The QR addressed in this cell are related to the events that trigger QR, and the QR related to the functions involved in the business scenario. The QR related to events triggering QR are represented by the same QR elements included in tactical goals; for example, increase security of an account when authentication fails. The QR related to the functions involved in the business scenario are represented by the same QR semantics included in tactical functions in cell (2,4) and may be flagged in the scenario model; for example, secure fund transfer when authorized access.
5.5.13 Operational Goal, Cell (3,1)

Operational goals are the operational form of goals assigned to an agent or system component for realization. This cell captures requirement goals in term of intent and the means to realize them. The refined goals and the means to realize them are best captured through tree structure notation to illustrate the types, level of refinement and contribution to different goals. This structure helps in making design decisions amongst different ways of realizing requirements. The SR model of i*, the KAOS goal model and the NFR framework for QR are examples of such notations. The QR addressed in this cell are related to different means of realizing the refined QR goals. The refinement and contribution links facilitate selection of decisions among different alternatives. The QR elements that express the different means to realize QR include QRID, mean, description, parent QRID, assigned stakeholder or functional component and the contribution link expressed in the tree model; for example, the QR goal QRID 1.1 confidentiality, expressed in cell (2,1) is operationalized by 1.1.1 authorisation with UN/PW, 1.1.2 biometrics and 1.1.3 access reader, the three different ways of realizing security account information.

5.5.14 Operational Agent, Cell (3,2)

The operational agent cell represents the agents responsible for realizing the operational goals and the interaction points between different types of agents and system resources. It captures who is responsible for the operational goals. Agent responsibility notations can capture the instant level responsibilities to realize the operational goals and the interfaces for interaction between agents and the system, e.g, the agent responsibility model. The QR addressed in this cell are related to the performance level of the agent who is responsible and the skill level required for realizing these operations. The description elements that express the QR of this cell are the same as those specified in the operation goal cells; however, the topic refers to the agent performing them; for example, a credit agent responsible for validating account access details to ensure secure access to an account. The operational constraints related to the agent skill level for a specific task are expressed through flagging agents with the required skill level; for example, a credit agent must comply with payment services regulations [147] which certify the credit agent as a secure body.

5.5.15 Operational Structure, Cell (3,3)

Operational structure represents software and environment entities and interrelations between them to realize the operational goals. This cell captures entity definitions of domain properties used by the goal, agent, operation and behavioural
models. The notations detail system and environmental entities for data, resources and agents including attributes, functions and relationships captured in UML class diagrams or ER diagrams. The QR addressed in this cell are related to one of two concepts: the performance constraint for accessing, retrieving or constructing an object, or the object that the controls performance constraints of an operation. The performance constraints are represented in the same QR description elements included in the operational goal; however, the topic refers to an object, for example secure access to account funding information. The control relationship is present when an object controls the performance of system behaviour; for example, the login object controls the 3.1.3 time to access customer service.

5.5.16 Operational Function, Cell (3,4)

The operational function cell represents the detailed functional units selected to realize operational goals with their interrelations and related domain variables. This cell captures the functional units, input/output variables, any environment pre/post-conditions and the agent responsible for realizing the function and interrelation among the functional units. The notation expresses the functional model, which may be a dynamic view including data flow representing the input and output of functional components; or control flow representing connections between functional components. The operational model and UML use case diagram are examples of notations used to model this view. The QR addressed in this cell are related to the expected performance of a functional unit and the performance constraints controlled by the unit. The performance measures of operational function are represented by the same QR description elements as for the operational goal cell; however, the topic refers to a function, for example, reliability of reports generated by ATMs. The control relationship can be expressed when a function controls the performance measures of a system; for example, the staff rota scheduling function controls the 3.1.2 response time to respond to a customer enquiry.

5.5.17 Operational Architecture, Cell (3,5)

Operational architecture captures the physical distribution of system objects identified at the operational level including data, agents, functional components and communication links among them. This cell captures the physical locations of system components and objects, and also the communication links, interfaces, access and information exchanged. The UML communication diagram, architecture diagram and component model are examples of notations that capture the architecture and the interfaces between system components. The QR addressed in this cell are related to the
performance of communication links and access interfaces among system components. The QR description elements that express the QR related to operational architecture are the same as those included in the operational goal cell; however, the topic refers to the interface communicating between different objects; for example, the integrity of account information when accessed through services channels, ATM, online, phone and customer service desk.

5.5.18 Operational Behaviour, Cell (3,6)

Operational behaviour captures a concrete description of system scenarios and the related environment and operational properties. This cell captures when to trigger operational system components and changes in the system properties following those operations. System behaviour is captured by the flow of system operational components such as UML activity diagrams, the flow of operation between agents such by UML sequence diagrams and the object behaviour transition by the UML state transition model. The QR addressed in this cell are related to the events triggering the QR and also the QR related to the functions, object and agent involved in the system scenario. The QR description elements related to events that can trigger QR are the same as those for operational goals; however, the topic refers to an event, for example, change account security when a user enters three wrong passwords. The QR description elements related to system scenarios are the same as the QR elements included in operational object, function and agent, flagged in the scenario model; for example, secure fund transfer between accounts when authorization is by an account holder.

5.6 The QRMF Contributions and Features

The contributions of this chapter are summarized in the following six points. The first point is offering a QR description form which explicitly defines the elements to identify, analyze and validate QR. The second point is the introduction of QRID within the QR description form, to facilitate tracing the evolution of a QR and the interrelation with other QR. The third point is the design of the QR description schema which depicts the syntax linking the elements of QR description form and the interrelation among them; this schema can be easily adapted to develop a QR modelling, analysis and validation tool. The fourth point is the framework's representational schema, which has two generic dimensions: the rows represent the transformation of requirement levels from strategic to technical phases, and the columns represent the six requirement modelling views [80]. The intersections between row and column depict a unique modelling perspective of
requirements with the related QR; in addition, putting cells together forms a complete requirements model reflecting the development of requirements with the related QR.

The QRMF features can be outlined in the following four points:

- The framework offers a structure to express QR and organize modelling perspectives. This is demonstrated in the QR description form and the representational schema. The structure plays a significant role in constructing requirements models systematically, and therefore can guide requirement engineers to construct comprehensive and consistent models.

- The framework's representational schema is based on two generic concepts of RE. The rows reflect the flow of requirement development phases characterizing the knowledge of early and late phases of requirements. The columns represent the entire requirement modelling views. The framework can make use of various requirement modelling approaches and modelling notations.

- The framework offers a complete model of QR in two ways. The first is by repressing the elements related to identifying, analyzing and validating QR in the QR description form for early and later requirement phases. The second is by addressing these concepts from the different modelling perspectives of the framework's representational schema. Although this schema is described as generic, it is complete as it reflects all the RE modelling views and the main levels of requirement development.

- To avoid limiting its flexibility in adopting different RE approaches and modelling notations, the framework does not explicitly define the relationship between the different modelling perspectives introduced in the framework representational schema. However, the framework offers a traceable description of the QR through introducing QRIDs by which the QR can be traced throughout the evolution of the requirement models.

5.7 Summary

To construct a complete model for QR, additional effort is required to address the complexity and diversity of QR concepts. This chapter addressed this challenge by introducing a multi-perspective modelling framework, the QRMF. The framework can guide requirements engineers to integrate QR with requirements models. In order to achieve this goal, the first objective was to identify the QR concepts related to the requirement tasks of the software product, as illustrated in section 5.3. The second
objective was integrating the identified QR with the related requirements models in a two-dimensional schema, illustrated in section 5.4. Section 5.5 presented a detailed description of the modelling perspective of each cell of the framework schema. Finally, section 5.6 detailed the contributions of the framework and its features.
6 The QRMF Meta-Model

6.1 Introduction

The multi-perspective framework for QR modelling introduced in Chapter 5 enables expression of the QR related to the six requirements modelling views: goal, agent, structure, function, architecture and behaviour. Although the framework's representational schema has been elaborated in an abstract form, a higher level of abstraction at meta-meta-level is essential to capture the framework's features in a comprehensive and generic manner. A higher level of abstraction aids comprehension of the features, further guides the modelling process and is also a reference blueprint for developing a modelling tool applicable to the framework. This chapter introduces different modelling levels for the QRMF to aid understanding its abstract concepts. Four abstraction levels are described in section: 6.2 Level 3 is the highest abstraction level and Level 0 the lowest, a concrete instance of a running system. This chapter focuses on explaining the meta-meta-model of the QRMF, level 3, as the other modelling levels are covered in the rest of the thesis. Section 6.3 presents an overall meta-meta-model that integrates the modelling views and related QR concepts. Each of the sections from 6.4 to 6.9 describes the meta-meta-model of a specific view with the related QR concepts. Finally, section 6.10 provides a summary of the chapter.

6.2 QRMF Meta-Model Structure

A meta-model is a domain-independent conceptual schema at an abstract level which includes concepts, relationships, attributes and constraints of a specific model [80]. The conceptual abstraction determines and guides construction of lower-level models related to specific domains or application settings. The QRMF offers four modelling levels: the meta-meta-model (Level 3), meta-model (Level 2), model (Level 1) and instance-model (Level 0). The two meta levels play a significant role in explaining the QRMF, which is composed of multi-perspective modelling concepts and also provides a solid base for adopting the framework. The meta-models proposed in this research are structural and hierarchical. The structure demonstrates different views of requirements models and the relationships between them. The hierarchy explains the multi-perspectives modelling framework. The meta-level models also demonstrate the QR concepts relevant to each requirement view.
It is important to recognize the four different levels of the QRMF to understand the context of meta-level modelling, before the details of the meta-meta-model are elaborated in the remainder of this chapter. Figure 22 illustrates these modelling levels and the hierarchical relationships between them.

Figure 22 The Meta-Modelling Levels of QRMF

6.2.1 The Meta-Meta Level

This level refers to domain-independent abstraction. It offers a high-level description of the concepts introduced in the QRMF representational schema. This higher level of abstraction groups together the various modelling concepts of the framework's modelling views introduced in the schema. The meta-model is made up of six core meta-concepts that abstract the requirement modelling views, the meta-relationships that link those meta-concepts, and meta-attributes of the meta-concepts or meta-relationships. Level 3 of the QRMF is illustrated in the top level of Figure 22. The package diagram in the middle demonstrates the overall QRMF meta-model outlining the main concepts and relationships as detailed in section 6.3. Each of the packages is related to the meta-concept of one requirement modelling view, separately detailed in sections 6.4 to 6.9.
6.2.2 The Meta-Level

This level also refers to domain-independent abstraction. It details the meta-concepts composing the framework models to guide constructing domain models using the framework. The level 2 meta-model is illustrated by the two-dimensional matrix in the second level of Figure 22. This meta-model was described in the previous chapter as the QRMF representational schema. Each of the meta-concepts in level 3 refers to higher-level concepts of one column of the representational schema. The intersections between the framework rows and columns define the meta-concepts of a single modelling perspective.

6.2.3 The Domain Level

This level refers to the modelling concepts of a specific software system in a specific domain; for example, a bank transaction system and a healthcare radiology reporting system. A system model is structured from the domain-level concepts according to instances of the corresponding meta-concepts and meta-relationships inherited from meta-level 2. Hence, at this level models are considered as an instance of the above level (the meta-model/representational schema). For example, a use case model is an instance of concepts describing the tactical function in cell (2,4) of the QRMF representational schema, illustrated in level 2 of Figure 22. Chapters 7 and 8 describe the domain-level model, level 1 of the framework, using two system examples of the banking domain and healthcare domain, respectively.

6.2.4 The Instance level

This level refers to a specific instance of the domain-level concepts in the running system, shown as the lowest level 0 of Figure 22. For example, a transaction number 1234 of fund transfer between account A and account B is an instance of the fund transfer function of the operational model described in cell (3,4) of level 1. This level is not included in the thesis as it related to the software runtime concept which is outside the scope of this research.

6.3 QRMF Meta-Model

This section describes the overall structure of the QRMF meta-model. The meta-model offers an initial integration mechanism by defining and interrelating the key concepts of all requirements modelling views at a meta-meta-level, level 3. The meta-model is made up of the six requirement view meta-concepts, meta-relationships between these concepts and the related QR meta-concepts. These level-3 concepts are represented at a high level of abstraction using the UML package diagram to provide an overall
structure and to simplify the readability of the meta-model. Figure 23 shows the overall meta-model structure where the meta-concepts are represented by packages and the meta-relationships by the links among them. The requirements meta-concept includes goal, agent, structure, function, architecture and behaviour meta-concepts. The meta-relationships among the meta-concepts are depicted by an association link that describes the static relationships for integrating the meta-concepts of RE modelling views. The remainder of this section briefly describes the meta-concepts included in those packages, the meta-relationships linking them and the related QR meta-concepts.

![Figure 23 An Overview of the QRMF Meta-Models Representing Requirement Views](image)

### 6.3.1 Goal package

This resembles the generic meta-concepts of the goal modelling view of RE. The goals simplify an agent’s demand to achieve through the system, a goal linked with <<demand>> association. A goal is normally refined into multiple specified goals called sub-goals until operational goals are reached. The sub-goals have the same parent-goal features and are therefore linked with the <<refine>> association to the same package. A leaf-goal, located at the lowest level of refinement, is operationalized by a function linked with <<operation>> association or assigned as an agent responsible for realizing it linked with <<responsible>> association. The QR meta-concept is a type of the goal meta-concept, called softgoals, it refers to an agent’s demand to satisfy, refined into QR sub-goals, operated by functions or assigned as an agent’s responsibility. A detailed goal meta-model is further described in section 6.4.

### 6.3.2 Agent package

This resembles agents interested in the system goals and agents responsible for realizing the operational goals. An agent meta-concept either refers to an agent demanding goals, which is part of the goal modelling view, presented with <<demand>>
association; or it refers to an agent responsible for realizing an operational goal, which is the agent responsibility modelling view, presented with <<responsible>> association. An agent can be a human who is responsible for undertaking a function or a system component that performs a function, presented with <<perform>> association linked with the function meta-concept. The meta-model discloses the distribution of agents as part of the architectural model, presented with <<located>> association linked with the architecture meta-concept. The QR meta-concepts related to the agent meta-concept are the QR of agent operational concerns, such as agent availability and agent skill level; and also the QR for agent performance level such as response time and reliability. The QR meta-concept related to agent demands is included as QR goals in the goals meta-concept. The detailed agent meta-model is elaborated in section 6.5.

6.3.3 Structure package

This resembles the structure or object modelling view of RE. The meta-model defines the conceptual object and domain properties used by the functional modelling view, presented with <<using>> association linked with the function meta-concept. The meta-model discloses the distribution of objects as part of the architectural model, presented with <<located>> association linked with the architecture meta-concept. The QR’ meta-concepts related to the structure meta-concept are the interface performance to access an object, performance level to construct an object instance and object constraints performance level. A detailed structure meta-model is elaborated in section 6.6.

6.3.4 Function package

This resembles the functional modelling view of RE. A function is an operational component that satisfies underlying goals. The function meta-concept captures functional features and their links with all the requirements meta-concepts including goal, agent, structure, architecture and behaviour through <<operation>>, <<perform>>, <<using>>, <<located>> and <<trigger>> association links respectively. The QR meta-concepts related to the function meta-concept are the QR of functional performance level and the performance level controlled by a function. A detailed function meta-model is elaborated in section 6.7.

6.3.5 Architectural package

This resembles the distribution and communication modelling view of RE. This meta-concept outlines a static structure of the physical meta-concepts distribution presented with <<located>> association links with agent, function and structure. The
communication among these meta-concepts depicts the dynamic behaviour of a system. The QR meta-concepts related to the architecture meta-concept are the QR of interface performance concerns between the meta-concepts, including access, communication and information exchange related QR. A detailed architecture meta-model is elaborated in section 6.8.

6.3.6 Behaviour package

This resembles the state transition or sequence modelling view of RE. This dynamic view of requirements refers to particular scenarios to be achieved or prevented by the system. The behaviour view of requirements is implicit within other modelling views [80], thus the relationship with them is also implicit. It is not possible to explicitly model all the system scenarios in graphical models, therefore the behavioural model focuses on the significant scenarios. The behavioural meta-model explicitly describes the behaviour meta-concepts that determine the sequence or transition that fires the meta-functions, presented with <<trigger>> association. The QR meta-concepts related to the behaviour meta-concept are the QR performance level of the concepts engaged in the scenario and the QR performance level triggered by an event. A detailed behaviour meta-model is elaborated in section 6.9.

6.4 The Goal Meta-Model

The goal is the objective a system is demanded to achieve. The goal model in RE represents the desired behaviours and expectations of different stakeholders interested in a system [14, 67]. It deals with the intentions of stakeholders that justify the rationale behind requirements and their operationalization. The goals are divided into two kinds: concrete goals, called goals, and softgoals. The goal is the objective an actor would like to achieve. The softgoal is a kind of goal that is satisficed, rather than satisfied [44], and is also called the QR goal that stakeholders demand to be satisfied.

Figure 24 depicts the structural meta-model of the goal model. The meta-model shows the constructs for expressing the key concepts of a goal model and the key meta-relationships that connect these concepts. The meta-relationships are expressed either as a label or a stereotype. The labelled link refers to a basic association between concepts depicted in the meta-model. The stereotype link refers to one of the relationship types specified in Figure 25. The stereotype is used to simplify the graphical representation of the meta-model and to give the meta-model room to include different relationship types. Referring to the meta-relationships through stereotyping implies that the flagged link can
be attributed to any of the relationship types specified in Figure 25. The refinement relationship is used when a parent goal is refined into one or more sub-goals. The sub-goals are specific goals or operational goals. The AND type is used when all the sub-goals take part in achieving the parent goal. The OR type is used when at least one of the sub-goals achieves the parent goal. The contribution relationship is used to express the contribution of the goal operation to satisfying the parent softgoal and other softgoals. The contribution can be of different types, but the most generic ones are help (to indicate support), break (to indicate harm) and natural (to indicate nothing or unknown).

The meta-concepts and meta-relationships of the goal meta-model are illustrated in Figure 24. The meta-concepts included are goal, agent, function and object, which are the key components to express the organizational context and the domain characteristics. The organizational context is expressed as the goal demand to be achieved, the agent that demands the goal, the objects to be provided and the function to be performed [14]. The goal meta-concept is the key concept in the goal model; it refers to the objective to be achieved or problems to be avoided. The softgoal meta-concept is a type of goal and is therefore linked with the goal meta-concept through a generalization relationship. The
generalization relationship implies that all the features of the goal meta-concept are inherent in the softgoal meta-concept. At an early stage of the requirements process, goals are specified as the agent's demand, as depicted through the demand association between the agent meta-concept and the goal meta-concept. This is true of the whole requirement process. However, goal refinement is another source to identify a goal, but with <<refinement>> the identified goals are specific. Goals can be refined at several levels before reaching operational goals, as demonstrated by a function meta-concept which can then be realized by an agent (either a system component or a person).

Although the softgoal meta-concept is a type of goal object, the softgoal requires greater detail, expressed through additional attributes and relationships. This facilitates describing QR as introduced in section 5.3.2. For example, type expresses QR terminology; topic expresses QR context, which can be object, function or agent; priority determines the significance of the QR to the agent's need; and expectation specifies the acceptable achievement level of the QR. The meta-relationships that link the softgoals are <<refinement>> and <<contribution>> as shown in Figure 25. <<Refinement>> is inherent in all the features of the goal meta-concept refinements. <<Contribution>> specifies the type of contribution to achieve the target goal. This relationship, together with the priority attribute, facilitates analysis and evaluation among alternatives means/operations to meet the parent goals or softgoals. This treatment of softgoals is a fruitful starting point for detailed softgoal structural refinement, analysis and evaluation.

Although the goal meta-model exhibits most of the RE meta-concepts, these concepts are declarative and not dynamic, which is why the architectural concept and the behavioural concept are not included. This does not weaken the goal model, which is fully devoted to the identification of relevant requirements and the rationale behind requirement specification.

6.5 Agent Meta-Model

An agent is an active component that can be a legal entity (person) or a software component. A legal entity includes a role, a person, a position or an organizational unit that demands system goals and influences requirement decisions. A software agent is a system component that performs a function that fulfils or contributes to the fulfilment of requirements; the legal entity may perform the same role. An agent model captures the distribution of responsibilities within the system; it defines system scope and
configurations in terms of the agent, interfaces and boundaries between software and environment; and finally, an agent model can be the basis of an architectural model.

The meta-model shown in Figure 26 depicts the meta-concepts of the agent responsibility model and the related QR meta-concepts. An agent influences the entire requirements models from the earliest requirement stage, from outlining the strategic goals that the agent demands to detailing the operations of each agent instance. The agent meta-model characterizes agent expectations, agent capabilities, agent functional performances that meet requirements assigned as responsibilities from the goal refinement model, and finally the agent hierarchy. The agent demand is depicted by <<expectation>> association. The meta-model depicts the responsibility of an agent to fulfil a leaf goal, shown by <<responsible>> association. The details of agent demands and responsibilities are expressed as in the previous section.

![Figure 26 Agent Meta-Model](image)

Figure 26 Agent Meta-Model

An agent meta-concept can be of different types, including department, actor, position and role. These types are depicted in inherent association among meta-concepts. The department-meta-relationships between them are depicted using association links. The department meta-concept demonstrates an organizational or external organizational unit that has a number of actors. The actor meta-concept represents an individual entity that may be assigned to one or more positions. The position meta-concept represents an organizational responsibility that performs single or multiple roles. The role meta-concept represents an organizational duty performed by an actor. The meta-relationships formulate heuristics for responsibility assignment by capturing the dependency chain among agents for traceability analysis, and also as an input for the architectural model.
The meta-model depicts the architectural QR meta-concept, which represents the interface performance constraints between agents through an aggregation link between architecture QR meta-concepts and the actor meta-concept.

The meta-model depicts the capability association referring to the ability of an agent to monitor and control the attributes and associations of an object meta-concept. In other words, an agent instance can capture or change the value of an object instance; an agent instance can hold, establish or delete an association between two agent instances. The object meta-concept depicted in this meta-model refers to the object interfaced with an agent; the QR related to this interface is represented by the architectural QR meta-concept aggregation with the agent meta-concept.

The meta-model also demonstrates a set of functions performed by an agent to meet its responsibility. This is represented by <<perform>> association linking the agent meta-concept with the function meta-concept. The functions performed by an agent provide the means to control agent capability over an object meta-concept and meet the assigned goal. The functional QR meta-concepts related to the function meta-concept are the performance level of the function performed by an agent, shown by a composition link.

6.6 Structure Meta-Model

The structure model, also called an object model, represents a static model of system. This model plays a significant role in requirements development as it defines concepts and domain properties used by all the RE models. The objects model forms a common vocabulary which can be referenced as a glossary of terms used constantly throughout the modelling and development stages [80]. The meta-model shown in Figure 27 depicts the meta-concepts of the structure model and the related QR meta-concepts.

The meta-model is composed of the object meta-concept, which is characterized by attributes meta-concepts to capture the object's intrinsic features. This is shown with aggregation type association between the two meta-concepts. The object meta-concepts are connected through three types of association link: model specific, aggregation and specification. These associations characterize the relationship between meta-concepts instantiated from the structure meta-model. Each of these types comprises different types, attributes and features. However, the object meta-model shown in Figure 27 does not include them as it does not have any influence on the QRMF.
The object meta-concept has two types: an entity and an active entity. An entity is either independent, i.e. it can exist independently of other instances or passive, i.e. an instance can not control other object instance behaviour [80]. This type of object comprises the object QR meta-concept. On the other hand, an active entity, also called an agent object, is an independent and active object, i.e. an instance has individual behaviours captured by state transitions of its attributes and associations [80]. The active entity may refer to an actor of an agent responsibility model but, as the concept here reflects a static model, the QR meta-concept is related to agent skill QR which refer to an agent object in a static state. The association depicts an aggregation type to stress that the active entity meta-concept is not the exclusive container for the agent QR meta-concept; it can also associate with the agent meta-concept.

### 6.7 Function Meta-Model

The functional model explicitly captures functional requirements by focusing on the service the system should deliver in order to satisfy its goals. Functions can be manual or automatic. The manual function is performed by an active entity (human) agent while the automatic function is performed by a software component. The function meta-concept was briefly explained in the agent meta-model from the perspective of an agent performing function. This section focuses on the dynamic modelling view of functions, including input, output, triggers and environmental post- and pre-conditions. Figure 28 summarizes the meta-concepts to express the functional model.

The function meta-concepts are connected with the association meta-relationship. The association meta-relationship can be one of four types: link, generalize, include and extend. The link association is used to establish communication between agent, behaviour,
object and function meta-concepts. The other three types of association are used to represent association between the instances function meta-concept. The generalize association is used to categorize the same function meta-concept. The include association is used to indicate when a function meta-concept is to perform another function meta-concept. Finally, the extend association indicates when the associated function meta-concept is optional or triggered under predefined conditions. The aggregation between interface QR meta-concept and association meta-concept demonstrates the quality of the interface to communicate the meta-concepts in the function meta-model.

Figure 28 Function Meta-Model

Figure 28 illustrates the same structural relationship as in Figure 26, that outlines the meta-relationships between the agent meta-concept, goal meta-concept and function meta-concept. The function meta-concept includes further meta-relationships that depict the dynamic modelling view. The <<trigger>> meta-relationship between the agent meta-concept and function meta-concept defines the sequence between functions in the functional model. The aggregation between the QR meta-concept and agent meta-concept reveals the performance level of the function performed by the agent. The function meta-concept is linked to the leaf goal meta-concept through the <<operation>> meta-relationship, demonstrating functional realization of the operational goal; this relationship is shown in detail in Figure 24.

The functional meta-concept is linked to the object meta-concept by the input-output relationship. This relationship demonstrates the input as a function required to
perform a task and the output as a function delivered after performing the task. This meta-relationship also demonstrates the pre- and post-conditions of environment variables relevant to performing a function. The aggregation between the QR meta-concept and object meta-concept reveals the performance level of the interface in accessing an object.

A function might be triggered by an internal or external event. An internal event refers to the outcome of a function that triggers another function. This is demonstrated by association links <<communicates>> and <<connected by>> meta-relationships between a function meta-concept and association meta-concept. The sequence and conditions of triggering functions is defined by the association type, as explained at the beginning of this section. The aggregation between interface QR meta-concept and function meta-concept demonstrates the performance level of interfaces of communication functions. On the other hand, an external event occurs when an agent such as user request, or an environment condition such as time or increased backlog, triggers functionality. This is shown when the agent meta-concept or behavioural meta-concept <<triggers>> the function meta-concept.

The functional model can be represented by an operationalization diagram or UML use case diagram. The use case provides an overview of the functions that an agent performs. The use case modelling leaves out the underlying details of functionality and intentionality which are useful from an analysis point of view [80]. Nevertheless, it is widely used in requirements elicitation and in communications between stakeholders [27]. The operational model captures more details about operations including environment pre- and post-conditions, goal-specific conditions, agents that trigger or perform an operation and, finally, internal and external functional triggers. These details are far too complex to be modelled or comprehended in a case diagram. However, they are useful for requirement analysis at a later stage of the requirements process. The meta-model proposed in Figure 28 illustrates general concepts about functional modelling but does not limit the requirements engineer to a specific modelling notation.

6.8 Architecture Meta-Model

The architecture model depicts the global structure that lays out the distribution and communication of RE modelling concepts. These concepts include active entities (human agents), operational components (functions) and data and information sources (objects). Each of these is a basic concept of a specific RE modelling view. The meta-models that depict the structure of each was presented previously in distinct sections. The
The architecture meta-concept describes the distribution of physical locations of the requirement meta-concepts. An agent can be a role, position or department that performs or accesses the requirements locally or remotely. This is depicted by the <<located>> meta-relationship between an agent meta-concept and architecture meta-concept. The agent QR meta-concept is linked to the agent meta-concept with an aggregation meta-relationship referring to the interface performance level between agent and the system, which was also illustrated in the agent meta-model.

An object can be a data input/output located locally within the system database or located in a remote database server exchange according to requirements. Depending on the requirements specification, data might be allocated to several locations for redundancy or load distribution. This is depicted by the <<located>> meta-relationship between an object meta-concept and architecture meta-concept. The object QR meta-concept linked to the object meta-concept with aggregation meta-relationship refers to the object interface performance level to access and to construct an object, which was also illustrated in the structure meta-model.

A function is an operation to accomplish requirement activities that may take place within the system servers or be remotely accessed as an external service. Depending on the requirements specification, a function might be allocated to several locations for redundancy or load distribution. This is depicted by the <<located>> meta-relationship between a function meta-concept and architecture meta-concept. The function QR meta-
concept is linked to the function meta-concept by the aggregation meta-relationship referring to the function interface performance level to trigger and communicate functions; this was also illustrated in the function meta-model.

The above meta-concepts are connected through two types of relationship: data flow meta-concepts or sequence flow meta-concepts. The data flow is used to communicate data with a function or an agent. This is demonstrated by the <<input/output>> meta-relationship between that data flow meta-concept and the object meta-concept, then the <<communicate>> meta-relationship between the data flow meta-concept and the agent or function meta-concept. The sequence flow is used to trigger one or more functions, performed locally or remotely by a system function or an agent. The sequence is controlled by an event which is caused by a function or environment condition. This is demonstrated by the <<trigger>> meta-relationship between that sequence flow meta-concept and the function meta-concept, then the <<control>> meta-relationship between the sequence flow meta-concept and event meta-concept. The event QR meta-concept refers to the event that triggers performance constraint. The architecture QR meta-concept refers to the interface performance in communicating with or accessing the requirement meta-concept. This is demonstrated through aggregation of meta-relationships between the architecture QR meta-concept and the sequence flow and data flow meta-concepts.

6.9 Behaviour Meta-Model

A behavioural model is an interactive view of requirement models: it captures the behaviour of agents in temporal sequences and of objects in state transitions. A sequence model captures the sequence of incoming/outgoing events between agents in timeline order. A state machine model captures the behaviour of an object by specifying different states of an object in response to events. Unlike the other modelling views, which offer an abstract description, the behavioural model depicts a concrete description of a scenario. A scenario captures an example of behaviour to be fulfilled or avoided. Although this modelling approach covers partial behaviour of a specific instance, it facilitates detailed requirements for a specific scenario that is implicit in the other modelling views and, therefore, the behavioural model facilitates undertaking decisions about requirements such as event sequencing and distribution of responsibility among agents. These decisions are further developed by capturing the QR related to the modelled scenarios. The
behaviour meta-model with the QR meta-concepts related to the behaviour model is shown in Figure 30.

The event is an action that affects the flow of activities and object transitions of a specified scenario. The event meta-concept is the central concept in the behavioural meta-model; it triggers requirement meta-concepts related to sequence models and state transition models. The event meta-concept is used to model time, conditions, status and environment variables that trigger the communication between function, object and agent meta-concepts. The behaviour QR meta-concept related to the event meta-concept is the performance constraint triggered by an event. This is demonstrated by the aggregation meta-relationship between the behaviour QR meta-concept and event meta-concept.

The sequence model depicts the sequence of interactions among agent instances. Figure 30 demonstrates an event meta-concept <<triggers>> a function meta-concept <<perform>> by an agent meta-concept in the meta-model. An event is caused by an action performed by an agent or a function, where several events may arise following an action. This is demonstrated by the <<causes>> meta-relationship linking the event meta-concept with the agent and function meta-concepts. The event can then trigger another function or interact with another agent according to the temporal sequence specified in the scenario. This is demonstrated by the <<trigger>> meta-relationship linking the event meta-concept with the function meta-concept and the <<interaction>> meta-relationship linking the event meta-concept with the agent meta-concept. The temporal sequencing

**Figure 30 Behaviour Meta-Meta-Model**

The event is an action that affects the flow of activities and object transitions of a specified scenario. The event meta-concept is the central concept in the behavioural meta-model; it triggers requirement meta-concepts related to sequence models and state transition models. The event meta-concept is used to model time, conditions, status and environment variables that trigger the communication between function, object and agent meta-concepts. The behaviour QR meta-concept related to the event meta-concept is the performance constraint triggered by an event. This is demonstrated by the aggregation meta-relationship between the behaviour QR meta-concept and event meta-concept.

The sequence model depicts the sequence of interactions among agent instances. Figure 30 demonstrates an event meta-concept <<triggers>> a function meta-concept <<perform>> by an agent meta-concept in the meta-model. An event is caused by an action performed by an agent or a function, where several events may arise following an action. This is demonstrated by the <<causes>> meta-relationship linking the event meta-concept with the agent and function meta-concepts. The event can then trigger another function or interact with another agent according to the temporal sequence specified in the scenario. This is demonstrated by the <<trigger>> meta-relationship linking the event meta-concept with the function meta-concept and the <<interaction>> meta-relationship linking the event meta-concept with the agent meta-concept. The temporal sequencing
The state transition model depicts transitions among different states of an object and the events causing them. The event raises a transition of the object from one state to another state. This is demonstrated by the event meta-concept raise <<transition>> to the object meta-concept shown in the meta-model in Figure 30. The transition meta-relationship is regulated by the transition meta-concept, which characterizes the environment attributes of object transitions. These attributes can feature some of the QR goals related to the scenario, demonstrated by the aggregation meta-relationship between the transition meta-concept and the QR goal meta-concept. The object state can be of different types, including initial state, final state and middle state. This is demonstrated by the aggregation meta-relationship between object and state meta-concepts. The different states are demonstrated by the generalize meta-relationship of the state meta-concept into the initial, final, wait state meta-concepts. The object QR meta-concepts are explained in detail in the structure meta-model.

6.10 Summary

This chapter described the overall structure of the QRMF meta-level modelling: the meta-meta-level 3, meta-level 2, domain-level 1 and instance-level 0. The chapter then focused on the meta-meta-level, demonstrating in detail the meta-models related to different requirement modelling views introduced in the QRMF. The meta-model offers an abstract structure of framework modelling views and integrates them into a single meta-model with related QR meta-concepts. This meta-model provides an integration mechanism to enforce the structural consistency and completeness of different modelling views of the framework. This abstraction may increase understanding of the framework features, further guide the modelling process, and help in the construction of modelling
tools that adopt the framework. Furthermore, the generic and independent meta-model has the flexibility to adopt the framework using different RE modelling notations and approaches. The chapter also elaborated each of the modelling views with regard to the related QR meta-concept in separate meta-models.

The meta-models described in this section support the QRMF framework by playing the following roles:

- The meta-model offers a higher level of abstraction by placing the framework features in a comprehensive and generic form. This abstraction level is represented using a streamlined graphical notation, the UML class diagram. The class diagram is widely used by requirement engineers to represent the static structure of modelling concepts. Such a simplified, graphic and structural representation may aid the comprehension of the framework features and further guide the requirements modelling process.
- The meta-model offers a platform to ensure that the modelling views are consistently related and complement each other. Furthermore, this platform facilitates the tracing and checking of modelling concepts in different modelling views through the meta-relationships represented in the meta-models.
- The meta-model stresses the framework’s generality, reusability and flexibility features, delineating generic concepts related to different modelling views independent of specific modelling notations or approaches. Examples of the applicability of the meta-meta-model and generality of the framework are represented in Chapters 7 and 8. An evaluation of the meta-model’s compatibility with various modelling notations and approaches is introduced in Chapter 9.
- The meta-meta-model can be used as a modelling template to guide generation of models for each cell of the framework.
7 The QRMF Modelling Process

This chapter looks at the modelling process using the QRMF, which facilitates the construction of requirements models with related QR in a systematic, coherent and integrated way. Chapter 5 presented a variety of heuristics for each of the modelling perspectives in the framework, and Chapter 6 an abstraction of the concepts related to each of those perspectives at a meta-meta-model level. This chapter is concerned with building an entire requirements model out of these perspectives. It presents a constructive method for constructing a robust and consistent model of QR integrated with other requirements models. It is an incremental process to identify, analyze and validate requirements with their related QR in an integrated multi-perspective modelling view. The framework modelling process consists of three phases that reflect the development of requirements knowledge, represented in the framework matrix as rows. Each phase follows the process of identifying and expressing the concepts related to each cell on the row. The flow among the cells indicates the interrelation among the modelling perspectives. These interrelations either reflect possible data dependencies among the cells or suggest a sequence in the construction of the models.

The chapter begins with an overall description of the modelling process. The remaining sections follow the modelling successive phases. Each section explains the corresponding step, what it consists of and how it can be run. The method is shown in action in a simple bank transfer example to illustrate most of the issues raised in a typical requirement modelling process. Section 7.2 explains how to capture the requirements at the strategic phase, section 7.3 how to construct the requirements models at the tactical phase, and section 7.4 how to construct the requirements models at the operational phase. Finally, section 7.5 provides a summary of the chapter.

7.1 Overall Modelling Process for the Framework

The framework process consists of three generic phases that reflect the evolution of requirements knowledge. These phases demonstrate the framework's rows: the strategic, tactical and operational levels. As outlined in section 5.4, these levels do not reflect a specific process but demonstrate a generic form of knowledge developed throughout the requirements process. Each of the three phases consists of a number of steps to identify and express the concepts related to each cell in each row. Figure 31 is an overall view of the framework modelling process; the arrows between the three phases refer to the sequence flow of the modelling process.
The first phase shows how to capture the requirements relating to the strategic level. This phase conveys high-level requirements relating to the organization and business concern in a textual form. The modelling process does not suggest any chronological order between the cells but suggests gathering as much information as possible within each cell to better understand the requirements of the project.

The second phase shows how to capture the requirements relating to the tactical level. This phase offers high-level models of what the software is capable of doing. These models are constructed in two ordered steps that form two generic models to capture the tactical requirements. The first step is a static model of the tactical concepts representing the goal, agent, structure and function views of the requirements. The second step is an interactive model that demonstrates distribution and flow among the tactical concepts representing the architectural and behaviour views of the requirements.

The third phase shows how to capture the concepts relating to the operational level. This phase presents a detailed model of how the software will operate to realize the requirements. The models of each operational concept are expressed in a separate step. This phase has a generic order, with the static models being constructed before the interactive models. The information flow arrows only suggest the order among the steps, and by no means prescribe a strict sequencing among them.

Figure 32 is a blueprint of the framework modelling process detailing the three phases and the steps in each phase. The arrows are not interpreted as a strict sequence as many of the steps are intertwined and conducted in parallel, and backtracking may be required to check consistency among models which is represented with feedback loop. Sequence numbers for most of the steps are not specified, to avoid imposing an order on the flow whilst adopting different requirement modelling approaches. This also avoids limiting the flexibility of the framework. Before detailing each of these phases and the steps for constructing the related models, we emphasize that the modelling process offers generic guidelines on the use of the framework by adapting different requirement modelling approaches.
7.2 Phase One: Capture Strategic Requirements

This phase aims to capture the organizational and business requirements of the software project at hand. These requirements convey business details that determine the scope of the project, and identify its objectives and constraints. This phase is conducted at the start of the requirement process where a high level of detail can be elicited. Capturing all possible information relating to stakeholder expectations, business domains and important scenarios can help develop clearer details about requirements. The strategic details are typically captured through the requirements elicitation and domain understanding activities of the requirements process. Besides the requirement engineers, strategic stakeholders such as the top manager, CEO and CIO may be involved in this phase. Business language, normally based on textual formats such as preliminary draft proposals, tender documents, minutes of meetings, policy documents and so on, is used to express
the outcome of this phase. Various interactive techniques can be used such as interviews, observation, questions and brainstorming.

This phase aims to address the six strategic requirements views in the corresponding cells of the first row. There is no strict order in which the cells are selected: at this phase the information is generic, and a single requirement statement can relate to more than one strategic view. For example, secure fund transfer from ATMs answers why the system is needed, capturing strategic goals; QR goals of secure fund transfer answer how to achieve the goal; and the functions of fund transfer answer where the function is located, capturing the distributed architecture of the ATM. The means of expression depends on the information available at this phase. The remainder of this section illustrates how to elicit and express the information related to each cell. Each cell is addressed in separate, unnumbered steps, which may be undertaken concurrently.

The strategic goal step: cell (1,1) contains information related to the high-level organizational objectives of the system. These objectives are also called coarser-grained goals. The information in this cell illustrates the purpose of the system by capturing intentional keywords from all available information sources, including interviews, meetings, tender documents and policy documents. It is also possible to identify the agent requesting the goals or constraints which will be expressed in cell (1,2). The statements of intent identify goals that might be related to an object, function or architecture which can be can be expressed in cells (1,3), (1,4)and (1,5) respectively. The QR are also expressed as coarser-grained QR goals which can be identified through statements of intent or constraints. For example, secure account information refers to the QR goal in cell (1,1); the account information is an object of cell (1,3) and secure account information is an object QR. The information related to the QR description form in this QR goal statement includes: QRID 1, QR type: secure, and topic: account information.

The strategic agent step: cell (1,2) expresses information related to the stakeholders who demand goals. The information in this cell defines goals by capturing the keywords for the agent's name, role, position and department, and is derived from similar available information sources. For an agent demanding a goal statement, the agent is expressed in cell (1,2) and the goal in cell (1,1). The QR related to this perspective are QR goals demanded by agents, as also expressed in cell (1,1). Further information related to the QR description form can be expressed at this step: agent expectation, measures and
priority to satisfy. For example, the QRID1 secure account information, stakeholder: CIO, expectation: 99.99% security of account, and priority: very high.

The strategic structure step: cell (1,3) contains the main data, information or resources related to the organizational goals or business domain. The objects can be identified by capturing keywords of data or references in statements of intent or domain descriptions culled from the usual available information sources. The QR related to an object within a statement of intent reflect the QR goal shown in cell (1,1), where the object is expressed as a QR topic of part of the QR description form. As in the previous example of QRID1 secure account information, what needs to be secure is identified, which is the account information object expressed in cell (1,3).

The strategic function step: cell (1,4) has information related to key functions or services to achieve the strategic goals. The information in this cell concerns the achievement of strategic goals by capturing action statements from all agents in the system, and all the sources and materials previously mentioned. The relevant QR are the expected performance measures of a function, also expressed as a QR topic in the description form, if found in a statement of intent; for example, the reliable fund transfer, QRID 2, type: reliability, and topic: fund transfer, which is an action expressed in cell (1,4).

The strategic architecture step: cell (1,5) gives an outline of the desired architecture as it relates to the distribution of agents, objects and functions. The information in this cell indicates where these components are located. The architecture can be identified through capturing keywords related to geographical locations or the agent hierarchy from all the usual information sources. The QR related to architecture are found in the interface performance between components, which refers to the QR topics on the description form; for example, the QRID3 of reliable fund transfer between different banks expresses the distributed architecture and the demand for reliable communication between banks in cell (1,5) and the function of fund transfer in cell (1,4).

The strategic behaviour step: cell (1,6) identifies behaviour related to significant scenarios to be considered or avoided. A scenario expresses the interaction between components when an action takes place or when a problem can be avoided. Strategic behaviour can be decided by capturing sequences of interactions or conditions that identify a critical event from all the available information sources and materials, such as interviews, meetings and observation. The QR related to behaviour are performance constraints that can be triggered by an event, which refers to the QR topics on the
description form. For example, QRID 4, increase account security when suspicious action, type: increased security, and topic: account, refers to objects expressed in cell (1,3), while suspicious action refers to an event expressed in cell (1,6).

### Strategic Level

<table>
<thead>
<tr>
<th>Cell (1,1)</th>
<th>Capture strategic goals and QR goals step.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell (1,2)</td>
<td>Capture strategic agent and the QR goals demand.</td>
</tr>
<tr>
<td>Cell (1,3)</td>
<td>Capture strategic object and the QRs relate to the objects.</td>
</tr>
<tr>
<td>Cell (1,4)</td>
<td>Capture strategic function and the QRs related to function performance level.</td>
</tr>
<tr>
<td>Cell (1,5)</td>
<td>Capture strategic architecture and the QR of interface performance level.</td>
</tr>
<tr>
<td>Cell (1,6)</td>
<td>Capture strategic behaviour of scenarios and the QR trigger by events.</td>
</tr>
</tbody>
</table>

**Figure 33 Steps for Capture of Strategic Requirements Phase**

A summary of the steps in the strategic requirements perspective with the related QR is shown in Figure 33. Although this phase does not illustrated in graphical models, it is vital for guiding the subsequent RE phases. Furthermore, it outlines information that is specifically related to QR, such as QR goals, interface performance, policy and economic constraints. The QR are generic for the whole system and can be detailed in later phases. It is important to express as much as possible of these generic teams using the QR description form, introduced in section 5.3.2, to minimize the risk of ambiguity and to facilitate traceability.

### 7.3 Phase Two: Capture Tactical Requirements

This phase aims to express system requirements by identifying lower-level details than the strategic requirements of the previous phase. The system requirement refer to what the software will perform, including details of business concepts and interrelations. The lower-level details refer to a specific or finer-grained form of the coarse-grained concepts specified in the strategic phase. The high-level goals are refined into fine-grained goals that contribute to the higher-level ones. The strategic agents are assigned responsibilities to perform to meet the fine-grained goals. The strategic functions are decomposed into sub-functional units to perform them. The strategic objects are expanded to include tactical objects related to the refined goals and decomposed functions. The strategic architecture is visualized to demonstrate the distribution and communication between the tactical components. Finally, strategic behaviour is visualized to demonstrate the flow of interaction among tactical components.

Requirement engineers and some stakeholders at departmental level are involved in eliciting business requirements and business domain understanding. Modelling
notations are used to identify, express and analyze details at the tactical level, as well as to capture interrelations. Although this phase contains more detail than was found at the strategic phase, models are only used to outline them. The outline is important as it confirms features required by the stakeholder, before conducting the detailed technical models of the operational phase. The gradual development of the requirements models encourages examination and verification of the requirements as they evolve. Therefore, this phase outlines tactical requirement details in a minimum number of models, as most modelling notations express several tactical concepts. For example, the SR models of the i* framework depict the concepts and interrelations by demonstrating the goals demanded by agents, the functions to perform the goals, and the resources to realize the functions. Another example, BPMN, expresses the agent, function, architecture and behaviour views that underpin the tactical level.

The six tactical requirement views join together in this phase. The number and order of the models to be constructed differs, depending on the RE approach and the specification of the project at hand. However, we propose two generic steps to construct tactical models which can be utilized without restricting the modelling approach or notation. The first focuses on constructing a static model and the second on constructing an interactive model. The remainder of this section describes how to undertake the two steps, what tactical concepts are addressed in each of them, and the interrelations between them and the strategic concepts.

**Step 1: Express static model at the tactical level:** the static model describes tactical requirement concepts and the relationships among them. The static view is a descriptive model that identifies the tactical concepts including goals, agents, objects and functions. The tactical goal, cell (2,1), expresses the sub-goals generated through the refinement of the strategic goals identified in cell (1,1). The tactical agent, cell (2,2), expresses the demands and responsibilities of finer-grained agents. The finer-grained agents are identified through decomposition of the strategic agents identified in cell (1,2) into lower department-level agents expressed in cell (2,2). The tactical structure, cell (2,3), expresses data and domain objects by expanding the strategic objects identified in cell (1,3) with objects related to the refined goals identified in cell (2,1) and the decomposed functions identified in cell (2,4). The tactical function, cell (2,4), expresses the sub-functional units generated by functional decomposition of the strategic functions identified in cell (1,4). These basic relations, obtaining tactical from the strategic concepts identified in the previous phase, are illustrated in Figure 34. The diagram does not include
the relations between tactical cells as these depend on the modelling approach used to construct the conceptual model. An example of how to construct the conceptual model by adopting a goal-oriented modelling approach is demonstrated after expressing the basic relations stated in the diagram.

Figure 34 Step 1, Basic Relations to Obtain the Static Model of the Tactical Level

The tactical goals, also called sub-goals, can be identified through the strategic goals refinement process. The refinement process identifies lower-level goals that contribute to higher-level ones. There may be several levels before reaching the operational goals that can be assigned to be performed by a function or an agent. The lower-level assignment of an operational goal to an agent addresses the responsible agent in cell (2,2) and the functional assignment addresses functions in cell (2,4). The QR goals identified in cell (1,1) in the previous phase are also refined in the same way in cell (2,1). The SR of the i* framework, and the goal model of KAOS, are examples of modelling notations that facilitate goal refinement.

The tactical agents are also called responsible agents, and can be identified through the agent decomposition process. The agent decomposition process identifies lower-level agents which can be guided by a hierarchic structure or organizational chart. Such a hierarchy may identify the department, position, role or person responsible for performing a tactical objective. This process extends to strategic agents, identified in cell (1,2), to address the demands and responsibilities of departmental-level agents/finer-grained agents in cell (2,2). Another method to identify tactical agents through responsibility assignment is to realize the tactical goals identified at the lowest level of refined goals in cell (2,1). The latter method implies a strong relation between goal refinement and responsibility assignment. The QR concepts related to tactical agents are
expressed variously as QR goals reflecting agent quality demands or as the performance level of an agent responsible for operational goals. The SR model of the i* framework and the goal model of KAOS are examples of modelling notations that can derive the agent model from the goal model.

The tactical functions, are also called sub-functions, functional components or functional units, can be identified through a functional decomposition process. The decomposition process divides the high-level functions into smaller related parts. Functional decomposition can reveal the hierarchy of the business functions that comprise the business operation without including its flow. The functional components can be captured by asking how a function is performed. This question can be asked several times until a sufficient level of detail is achieved, whereby all the business operations are represented and the connections between the functional components are included [148]. A Functional Decomposition Diagram (FDD) can be used to identify functional decompositions. This process extends the strategic functions in cell (1,4) to address smaller manageable chunk of functions that the system should deliver, expressed in cell (2,4). Another method to identify the tactical functions is through the functions assigned to realize the refined goals in cell (2,1). The latter method implies a dependency between functional components and goal refinement, which is not the case if the object-oriented or process-oriented requirement modelling approaches are used. Each of the functional components may be assigned to a responsible agent (human or system) expressed in cell (2,2) and may require the availability of certain resources, expressed in cell (2,3). The QR related to the tactical functions are expressed as performance measures of functional components. The SR model of the i* framework and the UML use case diagram are examples of notations that can be used to model functional components with related agents and required resources.

Finally, the tactical structure model expressed in cell (2,3), also called an object model, refers to the data and resources that must be available to perform the refined goals identified in cell (2,1) or the functional components identified in cell (2,3). The tactical objects in cell (2,3) are extended from cell (1,3) to include the tactical objects and the links to goals or functions that utilize them. The QR related to the tactical objects are expressed as the performance level in dealing with identified objects. The SR model of the i* framework is an example of a modelling notation that captures the objects related to the refined goals and functional components.
The previous four sections have shown four possible starting points in constructing a static model at the tactical level. Here, we demonstrate an example of how to express the static model based on the goal-oriented approach using the i* framework. This approach has been chosen for demonstration as it includes all the stated concepts using a single framework. Beside the sub-goals and QR softgoals identified through the goal refinement process, the other concepts identified are agents, objects and functions. These concepts provide an illustration of their respective modelling perspectives. The SR model of the i* framework captures the tactical goals in cell (2,1) through refinement and agent expectation; the agent responsibility for tactical agent in cell (2,2) is captured by assigning lower-level goals to responsible agents; the functions to be performed for the tactical model in cell (2,4) are captured by performing lower-level goals; and, finally, objects to be maintained for the tactical structural model in cell (2,3) are captured by data and resources necessary to achieve refined goals. The process of expressing tactical concepts using the i* framework is shown in Figure 35.

The model in Figure 36 shows an example of applying the process expressed in Figure 35 to capture the static model of the tactical level using the i* SR model. The example illustrates part of fund transfer in the banking system. The model expresses the goal of conducting fund transfer and the QR goal of securing account information as softgoals, flagged with QRID1. The goal is realized by two functions: request transaction and perform transaction. The softgoal with QRID1 is refined into two softgoals: QRID1.1 secure information access, and QRID1.2 authentication. Each of these QR goals is linked to functions to realize them. The QRID1.2 authentication is realized by a 1.2.1 login function,
whereas secure information access is realized by QRID 1.1.1 check access detail, and QRID 1.1.2 increased security function. The model also expresses the functional components to realize the goals. For example, to access a fund the bank system needs to retrieve the fund details and display them on the account holder side. Objects are also expressed when the account information is saved and retrieved from the fund object. Finally, the model expresses the two key agents involved: the bank and the account holder.

Step 2: Express interactive model at the tactical level: the interactive model describes a dynamic view of the tactical concepts that breaks down into three distinct elements: the distribution of tactical components, the communication between them and the sequence flow of communication. The first two concepts reflect the architectural view in cell (2,5) and the third reflects the behavioural view in cell (2,6). The strategic architecture in cell (1,5) is visualized to demonstrate the distribution and communication between the tactical concepts illustrated in the static model. The strategic behaviour in cell (1,6) is also visualized to demonstrate the sequence flow among tactical concepts illustrated in the static model. At the tactical level, an overall interactive model can express both the structure behaviour views in a single workflow diagram, such as a BPMN and UML activity diagram. Figure 37 illustrates the process of constructing the interactive model based on the static model at the tactical level and the basic relations elicited at the strategic level. The outline description of the business architecture and the required architecture expressed in cell (1,5), together with the outline descriptions of strategic

Figure 36 An Example of a Static Model of Tactical Level Using SR of i* Framework
scenarios expressed in cell (1,6), are visualized in graphic models to demonstrate the distribution and communication of the tactical concepts captured in the static model.

![Diagram showing the distribution and communication of tactical concepts](image)

**Figure 37 Step 2, Basic Relation to Obtain the Interactive Model at the Tactical Level**

Here we demonstrate an example of how to express the interactive model using a business process model with BPMN. This notation was chosen for demonstration as it includes sufficient detail for the structural and behavioural views in a single diagram that is easy to comprehend by an audience at the tactical level. The BPMN model describes the flow of activities within and across organizations. The architectural view that refers to the distribution of tactical concepts, including agent, object and function, is demonstrated through the swimlane. The swimlane groups modelling elements by representing organizational aspects which are denoted in business process diagrams by swimlanes via a two-level hierarchy. Pools represent organizations or participants. Lanes are sub-partitions of the pool that represent organizational entities. The communication part of the architectural view and the sequence flow of the behavioural view are demonstrated through connecting objects. The sequence flow specifies the order for flow objects and a message flow to describe the flow between business partners in different pools. Event and gateway controls establish the order of interactions, which reflects the behaviour of business scenarios. This structure allocates all the tactical concepts within the swimlane where the agents identified in (2,2) are expressed as pools or lanes; the objects identified in (2,3) are expressed as data objects or message objects; the functions identified in (2,4) are expressed as activities. The core BPMN model notation is sufficient to capture both architecture and behaviour views at the tactical level.
Figure 38 An Example of an Interactive Model of Tactical Level Using BPMN

The model in Figure 38 depicts the process expressed in Figure 37 applied to capture the interactive model of tactical concepts in the same fund transfer example, using BPMN. The architectural views are expressed by placing the agents, their functions and objects in separate pools. This shows that an account holder is external to the bank system. The communication between agents is expressed by message flow, the dotted lines, which also express a part of a sequence flow related to a behavioural concept. The flow is also expressed by a sequence flow controlled by gateways that direct the flow of activities and data. The overall process model expresses two business scenarios demonstrating the action to undertake when account access is granted or denied. The model expresses the interface QR relating to communications between agents such as QRID1.1 secure access, QRID2 compatible and QRID3 informative communication. The example also expresses a behavioural QR which shows a QRID1.1.2 increase in security level in the event of a failed login. The BPMN shown expresses all the tactical concepts apart from the goal concept. However, the scenario expressed in this interactive model demonstrates the fund transfer goal.

7.4 Phase Three: Capture Operational Requirements

This phase aims to express the operational requirements and explain the technical details needed to realize them. The phase transforms the business requirements expressed in the previous two phases into a technical view related to the software product. It extends the primary structure expressed in the previous phases with more specific
details relating to the software product and its operational environment conditions. This detailed view of requirements is constructed by requirement engineers with some stakeholders at the operation level through requirements analysis, negotiation and validation activities. Modelling notation is the chief language used because of its graphic ability to capture, express and comprehend the diverse details relating to operational concerns. The models used should provide details relating to the requirements view of each cell in the operational row. The models and documents produced during this phase should express sufficient detail to guide the remaining software development phases.

The requirements captured in this phase are more specific as they express lower-level details than those of the tactical concerns, and therefore each of the framework cells is expressed in separate steps to focus on the respective requirement view. The proposed order among the steps in this phase follows that used in the previous phase, where the specification of the static models is conducted before construction of the interactive models. This order facilitates capturing the operational concepts before expressing the dynamics of their communication. It was adopted to facilitate capturing QR details commonly neglected in the notations used to construct interactive models. The steps are not numbered, to maintain flexibility with different requirement modelling approaches. Figure 39 illustrates the suggested flow among steps to construct the operational model. The arrows refer to the information flow among the models in each cell. However, the arrows should not be interpreted as a strict sequence as many steps are intertwined. Backtracking may be required to check the consistency of the models.

![Figure 39 Steps to Construct Models of the Operational Level](image-url)
The remaining parts of this section illustrate how to capture and express the models relating to each cell. Each cell is addressed in a separate step to demonstrate the concepts captured, using the bank transfer as an example.

**Express the operational goals step in cell (3,1):** This step enables identification and capture of the operational goals, both hard-goals and QR goals, to realize the system. The operational goals are lower-level goals that can be assigned to be performed by an agent, a function or architectural setting. The operational goal models normally reveal several operational alternatives where a decision among them needs to take place for realization. The goal model with the selected operational options can be a reference model for constructing the other models in the operational concern row. To construct the operational goal model, the following activities may involve:

- Ensuring that all the goals and QR goals identified in cell (2,1) are refined until they express all possible alternatives for goal operation; further refinement may take place. The operational goals are then assigned to an agent or a function to perform them.

In practice for the QR goals, the following additional QR goal analysis activities need to take place:

- Specify the contribution links which indicate the positive or negative influence of the operational alternatives to realize the goals.
- Identify conflicts among goals and their operations.
- Undertake selection decisions among possible operational alternatives. The decisions are guided by the contribution links with the QR goal priority.

The SR model of the i* framework or the goal model of KAOS can be used to model the first activity. However, we recommend the SR model as it provides the notation for modelling QR goals, as softgoals. Then the NFR framework may be used, as this framework facilitates comprehensive treatment of the softgoals/QR. All three subsequent activities can be conducted through SIG of the NFR framework. Once the above activities are conducted, the resulting models can be a reference point for constructing the other requirements models.

The first activity of this step is skipped in demonstrating the bank transfer example as it is very similar to that in Figure 36. The demonstration for softgoal analysis activity is illustrated in Figure 40. It may be observed from Figure 36 that QRID1 secure
account information is refined into two sub-softgoals: QRID 1.2 authentication performed by the account holder and QRID 1.1 secure information performed by the bank. Figure 40 expresses three possible operational options for QRID 1.2 account access authentication: QRID 1.2.3 biometric, QRID 1.2.2 access reader and QRID 1.2.1 login. The contribution links indicate the opportunity for these operational options to realize QRID 1.2 authentication of account access and QRID 5 cost of an account. The contribution links indicate that authentication can be realized by any of the three options, although QRID 1.2.3 biometric will give a higher level of security. However, QRID 1.2.3 biometric option negatively influences QRID 5 cost of the account. This conflict needs resolution by a selection decision where the trade-off is between applying the highest security access option at the highest cost, and applying sufficient security access option at reasonable cost. This depends on the priority level given to QRID 1 security of an account as opposed to QRID 5 cost. The example shows that QRID 1.2.1 login for securing account access is selected to realize QRID 1.2 authentication and target QRID 5 cost.

Express the operational agent step in cell (3,2): this step explains how to construct the agent responsibility model. This model expresses the operational perspective of the agent model by exploring the alternative responsibilities of an agent at the agent instance level. In other words, the model represents the entire set of tasks performed by an agent. This step can be derived from operational goal models, especially when the i* framework is used to model the goal: i* conveys both agent-oriented and goal-oriented modelling approaches [14]. This stage may lead to the identification of new agents, human and computer, which are not identified in the goal model. Furthermore, the illustration of all the tasks performed by the agents may provide a base for constructing the functional model in cell (3,4). The QR addressed in this model reflect agent performance constraints and the agent skill levels required for the operational environment. These include user-oriented productivity concerns, such as response time, which become increasingly important in the definition of requirements for systems that provide direct production support. The following example demonstrates how to construct the agent responsibility model in the bank transfer example.
Figure 40 QR Goals Analysis Using SIG

Figure 41 Operational Agent Model Using Agent Responsibility Model

Figure 41 is constructed, based on the operational goal model including the selected operational options. The agents and their functions are initially included in the operational goal model illustrated in cell (3,1). The agents involved are illustrated in the SR model and the chosen operational options of QR goals are expressed in SIG. Further analysis of these agents and the responsibility assigned may introduce more specified agents. For example, the bank agent illustrated in Figure 36 refers to two agents within the
bank, the security server and the account manager server, as shown in Figure 41. The figure also demonstrates all the functions performed by an agent through grouping them together. The QR are marked with annotations attached to functions and agents. Those attached to functions rely on agent performance constraints to perform each function. For example, an account holder is given a session interval of 5 minutes to perform the transaction before re-login is required, where QRID 4 refers to the response time of the transaction. This satisfies security requirements by minimizing the risk of intrusive access. The annotations attached to the agents refer to the required skill levels needed to perform the assigned responsibilities in an operational environment. For example, the bank security server needs to be certified to ensure compliance with the security standard introduced in [147]; this QR is flagged with QRID 1.3 which implies that this QR is driven by the QRID1 of secure account.

**Express the operational function step in cell (3,4):** this step explains how to construct the functional model. It expresses the operational perspective of functional models by exploring the functional units, their input and output variables, related environment ‘pre- and post-conditions, interrelations that link functional units and finally the agent realizing these functional units. Some of the functions are expressed in goal models in cell (3,1) or the agent responsibility model in cell (3,2). Therefore we recommend constructing the functional model in accordance with these. The functional model can be represented by an operationalization diagram or, in a much more restricted form, by a UML use case diagram. The QR outlined in this step are related to functional performance constraints as well as performance constraints controlled by a functional unit. The following example demonstrates how to construct the functional model based on the agent responsibility and the goals model of the bank transfer example.

Figure 42 depicts the functional model using a use case diagram derived from the agent responsibility model and the goal model. Further analysis of these functions may introduce more specific functional units, or the conditions which give rise to them. For example, the figure illustrates that login failure raises the login monitor as depicted by the extend relationship. The QRID1.1.2.2 login monitor is one of the operational options of the QRID 1.1.2 increase account security. The figure also illustrates the sequence of linked functions with include relationships to realize QRID1.1 secure account access requirements, including QRID1.2.1 login, QRID 1.1.2.2 monitor login, QRID 1.1.2.1 apply encryption, and QRID 1.1.2.1 redirect to login. The QR are demonstrated with annotations linked to function, relationships linking the functions with an agent, and to system
boundaries. The QR attached to functions refer to the expected performance constraints of the functional unit. For example, the QRID 4 response time for check login is \( \leq 5 \) seconds. This period covers the time needed to conduct all the functions linked to the check login function, including monitor login, apply encryption and redirection to login. The response time annotated to these functions is part of the 5 seconds assigned for the login. The QR linked to the communication between agent and functional unit are interface QR, illustrated in detail in the operational architectural model in cell (3,5). For example, the access account information function should display the account information in a compatible format with a user access channel, annotated by QRID 2. The QR linked to the system boundary are global QR generic to all functions and based on the high-level QR goals demonstrated in cell (3,1), such as QRID 1 secure account information.

![Operational Function Model Using Use Case Diagram](image)

**Express the operational structure step in cell (3,3):** This step explains how to construct the structural model. This model expresses the operational perspectives of the structural model by exploring system objects, their attributes, their functions and relationships between objects. The objects resemble the information maintained in the system and the input and output of the system functions. The objects can be captured through the resources expressed in the goal model in cell (3,1) and through the data handled by the functions expressed in the agent responsibility model in cell (3,2) and
The structural model can be expressed using the UML class diagram or ER diagram. The QR expressed in this model act as performance constraints for accessing, retrieving or constructing an object. They also provide the control performance constraints of an operation.

Figure 43 depicts the structural model using a UML class diagram to demonstrate how to construct the structure model, in the bank transfer example. The model consists of four objects identified through the resources in the goal model, such as bank account, the data handled by functions in the agent model, and functional model data such as an account holder, access details and login backlog object. Further analysis of the identified objects facilitates the identification of different types of association linking these objects and also normalizes the overall object model. The QR linked to bank account objects, such as QRID1 secure account and QRID6 reliable transfer, refer to performance constraints for accessing and retrieving the object. The login-backlog object is identified to control the QRID1.1 secure access. Finally, the access detail object holds the data required for QRID1.2 authentication.

Express the operational architecture step in cell (3,5): this step explains how to construct the architectural model of the operational concerns. This model expresses the distribution of the identified system components including agents, functions and data, and the communication between them. These components are the ones captured in the conceptual models expressed in cells (3,2), (3,3) and (3,4). Their distribution and the communication between them may be based on the interactive model captured in the step 2 tactical phase. The QR expressed in this step are related to the performance of
communication links and access interfaces among system components. The UML communication model, activity diagram and business process models are types of notation used to express the architectural model.

Figure 44 depicts the architectural model using a communication diagram to demonstrate how to construct the architectural model in the bank transfer example. The model highlights the distribution of the three agents expressed in agent responsibility models in cell (3,2). The bank account manager server and the bank security server are two internal agents, i.e. part of the system, but the account holder is an external agent and is not part of the system. The association links indicate functions that communicate with these agents. The QR are shown with annotation links to express the interface performance constraints of communications. For example, account information must be displayed in a format compatible with the platform used by the account holder to access the bank system, which has been flagged with QRID2.

![Operational Architectural Model Using Communication Diagram](image)

**Express the operational behaviour step in cell (3,6):** this step explains how to construct the behavioural model, which model expresses the permissible system behaviours through scenario and state transition modelling to allow explicit illustration of system properties. A scenario can refer to a specific or an important system state or condition, so it is not possible to model all system scenarios through graphical modelling notations. The behaviour can be modelled through a sequence model to capture the behaviour of a specific agent instance, and a state transition model to capture a class of behaviours. The behaviour model can be based on the scenarios specified in the
interactive model captured in step 2 of the tactical phase. The QR addressed in this view are related to the event triggers performance constraint and the QR related to scenarios.

Figure 45 depicts a sequence diagram to demonstrate how to construct the behavioural model in the bank transfer example. The model highlights the sequence flow among the three agents expressed in agent responsibility models in cell (3,2) bank account manager server, bank security server and account holder. The functions expressed in the agent responsibility model in cell (3,2) and the functional model in cell (3,4) are demonstrated in the sequence flow of the scenario. The events are triggered when a function is completed and fires the next function in the scenario. The QR expressed in this example is related to an event that triggers QR to demonstrate the increase of the QRID1.1.2 account security when a user enters three wrong passwords.

![Figure 45 Operational Behavioural Model Using Sequence Diagram](image)

7.5 Summary

This chapter described the framework modelling process that guides the process of constructing models of the framework modelling views. The process consists of three phases that demonstrate the framework's rows: strategic, tactical and operation levels. Each of these rows consists of a number of steps detailing how to express the modelling view related to each cell in a row. The process depicts information flow among the models in each cell, which must not be interrupted as many of the steps are in a strict sequence; backtracking may be required to check the consistency among models. The process provides generic guidelines for constructing the models in the framework cells, using
different requirement modelling approaches and modelling notations. The chapter also elaborated each of the phases and steps to demonstrate what information is expressed in each cell, how this information is captured, and the interrelations with other cells. The bank transfer example was used in demonstration.
8 Case study: Applying the QRMF to a Radiology Reporting System

This chapter demonstrates the viability of the QRMF through the case study of a radiology reporting system. It describes the radiology reporting requirements and how the QRMF is used to construct requirements models that express complete and traceable QR. Through the case study we explore whether the framework presented in this thesis does add value to software development methods. Although case studies have no clear theoretical basis from which to draw strong conclusions, it is widely believed that a sound case study produces meaningful results in the software engineering domain [34]. The case study was conducted following the relevant guidelines for software engineering tools and methods [34].

The chapter is organized as follows. Section 8.1 outlines the case study hypothesis and the evaluation measures for validation. Section 8.2 provides background information about the radiology reporting system which was selected as a pilot project for evaluation. Section 8.3 presents the design considerations to minimize possible confounding factors. Section 8.4 explains how requirements models are constructed using the framework. Finally, section 8.5 presents an analysis and discussion of the case study results.

8.1 Case study hypothesis and the evaluation measures

The hypothesis for the case study states that the QRMF facilitates construction of complete and traceable QR models, where explicit annotations QR provide a better guide to addressing QR than do conventional requirements models. The hypothesis can be considered true if the following objectives are met:

- Demonstrate capability of the QRMF to construct complete and traceable QR models in a non-trivial, comprehensive and practical case study.
- Demonstrate how the models constructed using the QRMF offer a better guide to addressing QR.
- Demonstrate the ease of understanding the QR models through graphical annotation of QR from different RE modelling views.

In order to examine the validity of the hypothesis through these objectives, qualitative analysis of the use case models is conducted. The first objective is validated through three analysis activities, the first two to validate the completeness feature and the third to validate the traceability feature of the QRMF. The completeness feature is
validated through analyzing whether or not the conducted QR models address all the components in the QR description form; and also analyzing whether the QR are expressed from different requirements modelling views. The traceability feature is validated through analyzing whether upward and downward relationships among QR models are followed through the sequence of QRIDs. The second objective is validated through analyzing whether the constructed QR models offer more guidance than do conventional requirements models. The third objective, evaluation of comprehension, is typically conducted through audience feedback after using a proposed solution. However, in this research the evaluation was conducted in isolation from an actual audience, and therefore this form of evaluation could not be used. The alternative is to demonstrate how easy to read are the QR semantics addressed in the model, and how easy it is to trace the related QR using the QRID.

8.2 Background for the Pilot Study

The case study was selected from healthcare as this domain is under continuous pressure for service improvement to accommodate more patients, reduce costs and shorten treatment time with minimum risk to patients [143]. Most of these demands are QR related, and advances in information systems play a significant role in meeting them in such a complex environment [149, 150]. The complexity of the healthcare environment is due to the large number of stakeholders involved in a healthcare process, each with different interests and different levels of knowledge about the process [81]. To understand the context of this case study, including the variables influencing QR modelling, this section provides background information about the radiology process and a preview of the QR related to the stakeholders involved.

The main aim of radiology services is to take the required radiology images for patients and make them available with the radiologist's diagnosis reports. Radiology plays a critical role in patient care processes as it helps to diagnose patient illness accurately and in a timely manner. Advances in information technology have significantly improved the process of delivering radiology services [150]. For example, the emergence of computerized radiography has enabled the acquisition of all types of radiological images in digital form [151]; these digital images can be managed within and across a healthcare institution using advanced computer networks and distributed information systems. This advance has resolved many of the radiology problems, such as lost radiology films, limited space for archiving hard-copy films, and delay in communicating radiology information.
Furthermore, the radiology processes have reduced time and cost, with improved convenience and reliability [151].

Radiology services involve a chain of manual and computerized procedures across healthcare departments. Internal and external stakeholders are involved in tandem. Stakeholders typically interact in the following ways:

**Patients** are external stakeholders and the healthcare customers who want to be treated. Patients schedule a radiology scanning appointment, visit the clinic to schedule, follow the instructions to undergo the radiology scan, and finally wait for the radiology report after discharge from the clinic. Patients' QR can be summarized as timeliness, reliability and a convenient, patient-centered environment. Timeliness is characterized as the time taken to schedule an appointment, measured in days or hours, waiting time on arrival at the clinic for the scan, measured in minutes, and the time taken to receive the radiology report, measured in hours. Reliability refers to receiving the appropriate examination, avoiding unnecessary exposure to radiology doses, and receiving an accurate report. Convenience is characterized as a choice of appointment and location in scheduling; and respect for patient privacy and dignity during the radiology scan.

**Referring physicians** are external stakeholders and also customers of radiology services. Radiologists request the radiology procedure and view the radiology report to diagnose patients' condition. Physicians' QR can similarly be summarized as timeliness, reliability and patient-centred services. Timeliness is characterized as the time taken for reporting the results, measured in days or hours depending on the patient's clinical condition. Reliability is characterized as clear, accurate and well documented reports, relevant to the requested procedure. Physicians want an overall pleasant experience for their patients. The referring physicians may also be external providers of the radiology service, when consulted by the radiologist about specific cases. In this case, they demand reliable and timely access to patient radiology information: a high reliability rate to ensure access to the correct information, and timeliness in to accessing radiology images and information, measured in seconds.

**Radiology technicians** are internal stakeholders who have direct contact with patients. They explain the scheduled procedures, undertake the radiology scans, process the radiology images, check the image quality and discharge patients. The technicians' daily tasks are guided by dynamically updated worklists which detail the tasks to be conducted in sequence. The technicians interact heavily with radiology system
components, and their QR can be summarized as timeliness, reliability and efficiency. Timeliness is characterized as the time taken to access radiology images and information on the worklist, measured in seconds. A high reliability rate ensures accessing the right, updated information, and efficiency is characterized by minimum interruptions while conducting tasks, balanced distribution of workloads among themselves, and an easy to use system.

**Radiologists** are internal stakeholders who diagnose the radiology images, compile radiology reports and approve the reports. The radiologists’ daily reporting tasks are guided by dynamically updated worklists. They view the images and may also check patients’ medical history on the hospital information system or contact other physicians before writing the report. Radiologists are normally engaged with other duties such as teaching, research and consultation. Typing the radiology reports is considered an inefficient use of their time so they record the diagnosis which is then typed by a transcriber or a voice-recognition system. The reports are then verified and signed by the radiologist. Once a report is signed it is returned with the images to the radiology information system to be viewed electronically by the referring physician. In most cases, radiologists do not have direct contact with patients; only in some clinical cases are they involved in the scanning procedures of modifying appointments. For example, a radiologist can cancel a scheduled appointment if illness is diagnosed, or can schedule further appointments if more images are needed. The radiologists also interact heavily with the radiology system components. Their QR can be summarized as timeliness, reliability and efficiency. Timeliness is characterized as the time taken to access radiology images, patient information and worklist instructions, measured in seconds. Radiologists also demand timely responses from referring physicians if more information is needed, measured in minutes; speedy transcription time from transcribers, measured in minutes, or timely/instant voice recognition measured in seconds. A high reliability rate ensures accessing correct, updated information, is also demanded in the accuracy of human or system transcription of the report. Efficiency is characterized by minimum interruptions while conducting reporting tasks, minimizing consultation with physicians, balancing the distribution of workloads among themselves, and ease of use of the system.

**Radiology administration** takes care of administrative tasks such as inventory, payments, billing, etc. However, the core radiology service aside, other administrative tasks are outside the scope of this case study. The administration typically demands the conducting of radiology services in a timely, reliable, patient-centered, effective and cost-
efficient manner, that can readily accommodate more patients and satisfy their expectations without adding stress to the staff. Effective service implies providing a radiology service based on current scientific knowledge and best practice.

These stakeholder interactions and the flow of information are facilitated through integrated system components. The Integrating Healthcare Enterprise (IHE) technical framework [152] considers the components themselves as stakeholders, called actors, as they operate the departmental activities. The system components interact with each other and also with human stakeholders. The main features of these interactions are captured and summarized as follows:

**Radiology information system (RIS)** manages the workflow of processes in the radiology department by coordinating stakeholders’ activities and information flow. The RIS consists of a number of components. A worklist is the main coordination component, providing an up-to-date workflow that guides the radiology procedures at different stakeholder levels. The worklists are programmed to comply with certain policies or standards that reflect departmental goals. For example, a national radiology framework commits radiology services to an equal access policy with consideration of patients' clinical condition, by setting the following measures for reporting turnaround times: emergency patients are served first within 30 minutes; inpatients are served on the same day; and all other cases are served within the next working day [153]. The worklist is automatically updated as each activity is completed. This helps in accommodating emergency cases with minimum disturbance and greater communication, by assigning the emergency case to an available technician or to one about to finish the task in hand; and the same is true for modality, radiologists and other required resources.

**Picture Archiving and Communication System (PACS)** is a short-term archive that manages DICOM communication between modalities which involve image acquisition, post-processing and reporting. PACS provides an interface between the RIS and the archive.

**Archive** handles long-term data which can be accessed through PACS and is important for the responsiveness of the PACS system.

**Hospital Information Systems (HIS)** holds Patient Medical Records (PMR), interfaces with the RIS to send examination orders, access patient medical records, and
retrieve patient radiology reports. It also provides an interface with PACS for the display of patient images in other healthcare departments.

8.3 Considerations for the Case Study

The case study was undertaken in a research environment. Knowledge about the domain was developed through an industrial and scientific literature review about the radiology process. The framework modelling process is conducted by the author of this thesis. The description of typical processes employed in 21st century radiology systems was developed through study of [151, 154, 155]. The interactions among system components in radiology information systems were captured from the IHE technical framework [152], which describes system components as stakeholders but does not reflect real-word stakeholders of radiology services. The perspective of the stakeholders is very important in understanding this domain and its challenges, and in identifying the requirements for improvement; therefore, further literature [151, 154-162] was investigated. The core clinical and scientific aspects of the radiology field are not included in this case study because they depend on guidelines for best practice and do not anticipate further advances in the science and technologies of the field [163].

The process selected for evaluating the framework was introducing a Voice Recognition (VR) system into the Radiology Reporting Process (RRP) to achieve the goal of reducing the Reporting Turnaround Time (RTT) (time requirements). The focus here is on modelling time requirements to allow further elaboration within a reasonable number of pages, yet be rich enough to illustrate most of the issues raised in a typical model-based requirement process.

We acknowledge that many factors differ between conducting the case study in a research and an industrial setting. These factors are interconnected in contributing to the success of the framework; for example, the knowledge of requirement engineers and stakeholders about the requirement process, modelling notations and the domain. However, we focus in the case study on addressing the three objectives introduced in section 8.1 on QR modelling using the QRMF. This will allow us to identify merits or problems based on an empirical demonstration in the context of the case study. The investigation of causal relationships of all the interacting factors in the case study and the research are beyond the scope of this thesis.
8.4 Conducting the case study

This section describes how the QRMF is applied in the RRP by following the three phases of the framework modelling process introduced in Chapter 7.

8.4.1 Phase One: Strategic Level

The scope, objectives and constraints of this case study were captured through an intensive literature review, as described in the previous section. This rigorous review helped the author to identify the business and organizational concerns related to introducing a VR system into the RRP in order to shorten the RTT. Shortening the RTT is a key requirement in providing a high-quality imaging service. Extending the waiting time will harm the overall radiology turnaround time and impact negatively on other QR such as efficiency, and patient-centred, reliable care. The strategic concerns of the case study are expressed in plain text, listing the strategic requirement concepts with the related QR, as follows:

The strategic goal in cell (1,1):

- **Strategic Goal:** to use VR system in issuing radiology reports
- **QR:** this cell elicits softgoals

<table>
<thead>
<tr>
<th>QRID 1.</th>
<th>Type: time</th>
<th>Topic: reporting turnaround</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: reduce the time from saving the radiology scan to sending the report</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rationale: this QR impact other QR such as efficiency, and patient-centred, reliable care</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder: CIO</td>
<td>Priority: high</td>
<td></td>
</tr>
<tr>
<td>Expectation: reducing RTT to meet the 18 weeks treatment policy</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>QRID 2.</th>
<th>Type: efficiency</th>
<th>Topic: reporting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: conduct more reports with same time interval and same staff</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rationale: increase productivity and minimize cost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder: CIO</td>
<td>Priority: high</td>
<td></td>
</tr>
<tr>
<td>Expectation:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
QRID 3. Patient-centred worklist

<table>
<thead>
<tr>
<th>QRID 3.</th>
<th>Type: patient-centred</th>
<th>Topic: worklist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: decision to schedule the reporting task in the worklist based on patient’s medical condition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rationale: to serve critical cases first</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder: CIO</td>
<td>Priority: very high</td>
<td></td>
</tr>
<tr>
<td>Expectation: emergency patients are served first within 30 minutes, inpatients are served the same day, and all other cases are served within the next working day</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

QRID 4. Reliable report

<table>
<thead>
<tr>
<th>QRID 4.</th>
<th>Type: reliable</th>
<th>Topic: report</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: correct and readable report</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rationale: to avoid medical mistakes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder: CIO</td>
<td>Priority: very high</td>
<td></td>
</tr>
<tr>
<td>Expectation: 99.99% reliability rate</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Policies and constraints**: three policies with which the RTT must comply:
  - NHS policy for treatment time within 18 weeks.
  - Scheduling reporting tasks according to patients’ health condition, following national radiology framework policy for RTT measures [153] (emergency patients are served first within 30 minutes, inpatients are served the same day, and all other cases are served within the next working day)
  - A response time determined by other quality of care considerations.

- **Problems to overcome**: Typing backlog increase, increased load, absence of transcribers

The strategic agent in cell (1,2):

- **Strategic agent**: CIO, radiologist, physician, patient
- **QR**: this cell elicits strategic agents’ QR demands
**QRID 1. Reduce RTT.**

<table>
<thead>
<tr>
<th>QRID 1.</th>
<th>Type: time</th>
<th>Topic: reporting turnaround</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: reduce the time from saving the radiology scan to sending the report</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rationale: this QR's impact on other QR such as efficiency and patient-centred, reliable care</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder: CIO, radiologist, physician, patient</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Priority: high</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expectation: reducing RTT to meet the 18 weeks treatment policy</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**QRID 2. Efficient reporting**

<table>
<thead>
<tr>
<th>QRID 2.</th>
<th>Type: efficient</th>
<th>Topic: reporting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: conduct more reports with same time interval and same staff</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rationale: increase productivity and minimize cost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder: CIO, radiologist</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Priority: high</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expectation:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**QRID 3. Patient-centred worklist**

<table>
<thead>
<tr>
<th>QRID 3.</th>
<th>Type: patient-centred</th>
<th>Topic: worklist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: decision to schedule the reporting task in the worklist based on patient's medical condition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rationale: to serve critical cases first</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder: CIO, radiologist</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Priority: very high</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expectation: emergency patients are served first within 30 minutes, inpatients are served the same day, and all other cases are served within the next working day</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**QRID 4. Reliable report**

<table>
<thead>
<tr>
<th>QRID 4.</th>
<th>Type: reliable</th>
<th>Topic: report</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: correct and readable report</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rationale: to avoid medical mistakes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder: CIO, physician, patient</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Priority: very high</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expectation: 99.99% reliability rate</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**The strategic structure in cell (1,3):**

- **Strategic Object:** Scanned images, radiology report, worklist, PMR
- **QR:** This cell elicits the QR of the strategic objects
  - QRID 1. Reduce RTT, as introduced above.
  - QRID 4. Reliable report, as introduced above.
QRID 5. Accessible PMR

<table>
<thead>
<tr>
<th>QRID 5.</th>
<th>Type: accessible</th>
<th>Topic: PMR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: PMR is accessible remotely whenever it is needed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rationale: to compile reliable radiology report</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder: radiologist</td>
<td>Priority: high</td>
<td></td>
</tr>
<tr>
<td>Expectation: 24/7 accessibility</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

QRID 6. Clear scanned images

<table>
<thead>
<tr>
<th>QRID 6.</th>
<th>Type: clear</th>
<th>Topic: scanned image</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: clear and resizable image</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rationale: to conduct reliable radiology report</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder: radiologist</td>
<td>Priority: very high</td>
<td></td>
</tr>
<tr>
<td>Expectation: clearly viewed on 1024 × 768 monitor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

QRID 7. Responsive worklist

<table>
<thead>
<tr>
<th>QRID 7.</th>
<th>Type: responsive</th>
<th>Topic: worklist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: timely updated schedule of worklist duties based on the workload and patient's condition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rationale: to meet the target treatment policy. This QR directly impacts achieving patient-centred QR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder: radiologist</td>
<td>Priority: high</td>
<td></td>
</tr>
<tr>
<td>Expectation: immediate updates</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The strategic function in cell (1,4):

- **Strategic function:** radiology diagnosis, report writing with VR, report verification
- **QR:** This cell elicits the QR related to the strategic functions or services

  QRID 4. Reliable report, as introduced above.

  QRID 4.1 Reliable diagnosis

<table>
<thead>
<tr>
<th>QRID 4.1.</th>
<th>Type: reliable</th>
<th>Topic: diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: correctly diagnose the images</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rationale: to avoid medical error</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder: radiologist</td>
<td>Priority: very high</td>
<td></td>
</tr>
<tr>
<td>Expectation: 99.99% reliable diagnosis</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
QRID 4.2. Reliable VR report writing

<table>
<thead>
<tr>
<th>QRID 4.2</th>
<th>Type: reliable</th>
<th>Topic: VR report writing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: correctly type the dictated report</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rationale: to provide readable report and to avoid medical mistakes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder: radiologist</td>
<td>Priority: high</td>
<td></td>
</tr>
<tr>
<td>Expectation: 95.00% error-free spelling and 95% errorfree word matching</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

QRID 4.3. Reliable report verification

<table>
<thead>
<tr>
<th>QRID 4.3</th>
<th>Type: reliable</th>
<th>Topic: report verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: ensure the typed report correctly diagnoses the scan and is readable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rationale: to avoid medical mistakes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder: radiologist</td>
<td>Priority: high</td>
<td></td>
</tr>
<tr>
<td>Expectation: 99.99% reliability rate</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

QRID 2.1. Efficient report writing with VR software

<table>
<thead>
<tr>
<th>QRID 2.1</th>
<th>Type: efficient</th>
<th>Topic: VR report writing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: writing the report in minimum time and with minimum human intervention</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rationale: increase productivity and decrease costs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder: radiologist</td>
<td>Priority: high</td>
<td></td>
</tr>
<tr>
<td>Expectation:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The strategic architecture in cell (1,5):

- **Strategic architecture:** Integrate VR component with reporting component
- **QR:** This cell elicits the QR related to high-level architecture

QRID 8. responsive VR

<table>
<thead>
<tr>
<th>QRID 8</th>
<th>Type: responsive</th>
<th>Topic: VR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: type the report with same recording speed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rationale: to avoid any delay</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder: radiologist</td>
<td>Priority: very high</td>
<td></td>
</tr>
<tr>
<td>Expectation:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The strategic behaviour in cell (1,6):

- **Strategic behaviour:** To overcome an increase in the RTT due to increased reporting backlog, increased workload and a shortage of transcribers
- **QR:** This cell elicits the QR triggered by events related to the above behaviour

QRID 1. Reduce RTT, as introduced above
8.4.2 Phase Two: Tactical level

This phase expresses the business requirements based on the details captured at the strategic level and further investigation of the radiology domain. In this case study, a goal-oriented approach guided the construction of the tactical models. To conduct the first step that expresses static models, a strategic dependency model is constructed based on the strategic goal of cell (1,1). Furthermore, the strategic rationale model is constructed through goal refinement of the strategic dependency model and also with consideration of the elements elicited in the other static views (agent, structure and function) of the strategic level. In step 2 a BPMN diagram expresses the interactive model at the tactical level. This model is constructed based on the static models at this level and the basic relations of the interactive views elicited at the strategic level. The following two sections demonstrate the steps in conducting the radiology reporting case study.

Step 1 Express static model at the tactical level:

- i* explicitly captures QR as softgoals and facilitates decomposing them into possible functional alternatives. Figure 46 illustrates the Strategic Dependency (SD) model where the main goal is to introduce the VR component (goal) to reduce the RTT (softgoal with QRID 1). This QR is of a high level as it refers to the process of producing the report. The diagram shows that the CIO depends on the RIS to achieve these goals through the dependency link. This high level also illustrates other QR demands of the CIO: 2. efficient reporting refers to a function; 3. patient-centred worklist refers to an object; and 4. reliable report refers to an object, as depicted in cell (1,1).

- The static model at the tactical level is enriched by investigating possible ways to achieve the goals illustrated in Figure 46 with Strategic Rationale (SR) models. The information captured at the previous level by further domain investigation guides the construction of the SR model. This model illustrates detailed dependency among tactical agents and their goals, resources and functions. Further detail of the goals’ and softgoals’/QR’ operationalization is obtained using means-end relationships. Furthermore, the contribution relationship depicts positive or negative support of softgoal realization. Figure 47 illustrates three system agents (RIS, PACS, HIS) beside the human agents elicited at the strategic level. The goals of these agents are refined until the functions have been performed by each agent, the resources utilized and the related goals/softgoals achieved. For example, the model demonstrates that the function producing a report is performed by a radiologist, and is composed of seven sub-functions linked by composition relationships. The contribution relationship demonstrates that RTT is affected positively.
by worklist updates and VR report typing. On the other hand, RTT is affected negatively by consulting physicians instead of accessing PMR.

![Figure 46 Strategic Dependency Model for RRP](image)

![Figure 47 Strategic Rational Model for RRP](image)

**Step 2 Express interactive model at the tactical level**

The interactive model at the tactical level expresses the architectural and behavioural view with the related QR. Figure 48 depicts the interactive model using a business process model with the BPMN. The figure shows in a single diagram the main architectural and behavioural views for introducing a VR component into the radiology
reporting process. The architectural view refers to the distribution and communication of tactical concepts with the related interface QR. The agents are represented as pools or swimlanes with related functions as activities, and objects as data and events. The radiology department is represented in a pool which has three tactical agents (radiologist, RIS, PACS), each in a different swimlane. The physician and HIS are in different pools, which symbolizes external agents. The model illustrates that the PACS system is responsible for typing the radiology report with the VR component. Radiologists demand a QRID 8 responsive VR component to type the report at the same recording speed and to avoid any increase in PACS response time.

![Figure 48 Interactive Model for RRP in BPMN](image)

The diagram also depicts the time requirements of interfacing activities. For example, a radiologist waits for a physician for consultation, and a report waits for an available radiologist to approve it. The waiting time is longer when human agents are involved, which negatively affects the RTT; therefore, it is better to avoid this when possible. On the other hand, when machine components are interfacing, the waiting time is far less, depending on the machine’s capacity to process a request which is normally within milliseconds. Enhancing machine response times depends upon hardware components and is therefore excluded at this stage of modelling requirements. The model also depicts two other interfaces QR QRID 5 accessibility and QRID 6 clarity. The former implies 24/7 access to the PMR through the RIS to avoid any delay. The latter, clarity of radiology images, enables the radiologist to make diagnoses without needing to rescan, which would increase the RTT. Although the SR model depicted in Figure 47 illustrated these agents, their functions and allocated resources, the dependency link does not
illustrate the hierarchy of the architectural model, sufficient communication details between agents or the interface QR.

The overall process model in Figure 48 also expresses two business scenarios demonstrating the actions to undertake when producing or approving a radiology report. The behavioural view is depicted through communication links, events and gateways. It comprises a sequence flow to specify the order, data flow and gateway that feature the interactions of behaviour in business scenarios. The worklist schedules radiology tasks (reporting and approving) to achieve QRID 3 patient-centred worklist by considering patients’ medical condition; and QRID1 reduce RTT by considering the radiology referral date. This activity reschedules the worklist once one of the three activities (image/report checking, report dictation, and report verification) has been completed to achieve interface QRID 7 responsive worklist. Another interface is QRID 9 responsive consultation, when a physician’s advice is needed its QR for contacting a physician. QRID 8 responsive VR is also an interface QR to specify how the VR component is integrated with PACS.

QRID 9 responsive consultation

<table>
<thead>
<tr>
<th>QRID 9.</th>
<th>Type: responsive</th>
<th>Topic: consultation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description: physician can get swift consultation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rationale: to avoid delaying reporting</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder: radiologist</td>
<td>Priority: high</td>
<td></td>
</tr>
<tr>
<td>Expectation: consultation should schedule to meet the target RTT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8.4.3 Phase Three: Operational Level

This phase aims to express the operational requirements and explain the technical details with consideration to the operation environment. The phase transforms the business requirements captured in the previous two phases into a technical view related to the software product. The technical model of each requirement view is modelled separately in its respective cell to express sufficient detail for the remaining software development phases. This case study is conducted following the flow suggested in the modelling process introduced in section 7.4 of the previous chapter. The models in this phase can become very complex as they convey many interrelated details. To avoid this, we focus in this case study on modelling QRID1: reduce RTT of producing radiology report, and only related QR. The following sections demonstrate these steps.
Express the operational goals step in cell (3,1):

This step identifies operational goals (hard goals and QR goals) through goal refinement until it is operationalized by an agent or function. The goal refinement of VR report typing was shown in Figure 47. This model is a further refinement of the SR model shown in Figure 46, with a focus on the report production task and the QR related to QRID1 reduce RTT. The model shows that report production is conducted through two main tasks: image diagnosis and report writing. The diagnosis task is composed of four sub-tasks: access the assigned task, access the scanned images, access the PMR, and consultation. The reporting tasks are scheduled in the worklist object and performed by the schedule reporting task in the RIS. This task influences positively the QRID 3 patient-centred worklist and QRID 2 efficient reporting. The model shows that accessing PMR at HIS has a positive influence on QRID 1 reduce RTT, whereas consultation has a negative influence on it. Once the images are scanned, the report writing is performed through report dictation by the radiologist and report typing by the VR component integrated in the PACS. The VR report typing has a positive influence on QRID 1 reduce RTT and QRID 8 efficient writing, as it offers immediate typing from dictation without the need for transcribers.

Also in this step, the QR goals are analyzed further through SIG to indicate positive or negative influences of the operational alternatives and to undertake selection decisions among them. Figure 50 illustrates the SIG for reducing QRID 1 time [RTT] of the report writing task, which is refined into three softgoals: QRID 1.1 time [task access], QRID 1.2 time [diagnosis] and QRID 1.3 time [report writing]. Each of these goals is refined further until operation; for example, the QRID 1.1 time [task access] is realized through 1.1.1 monitoring [log], 1.1.2 schedule [worklist] and 1.1.3 assignment alert [164]. 1.1.1 monitoring [log] to schedule the task is based on an immediate update of the workload. 1.1.2 schedule [worklist] schedules the worklist based on the updated log and the patient’s medical condition. These two realization options influence positively QRID 3 patient-centred [worklist] and QRID 7 responsive [worklist]. 1.1.3 assignment alert [164] automatically informs radiologists about the next task instead of their having to enquire about it. An example of an operational option that negatively influences QR is also illustrated: QRID 1.3 time [report typing] is realized by 1.3.1 dictation with 1.3.2 VR typing [report] or 1.3.3 transcriber typing [report]. The figure shows that 1.3.3 transcriber typing [report] negatively influences QRID 1.3 time [report typing] and also QRID 2 efficient [reporting], and is therefore not chosen for operation.
Express the operational agent step in cell (3,2):

This step expresses the operational agent model and agent QR through exploring all the tasks performed by an agent at the instance level. Although the operational goal models illustrated in Figure 49 depicts the operational agents, it does not illustrate the
details of instance-level responsibilities. For example, in the agent responsibility model in Figure 51 two roles of PACS are illustrated: information management and VR report typing. The information management role is responsible for retrieving the scanned images timely and clearly and also for saving the recorded diagnosis and typed report. The other responsibility of PACS is to type the dictated report at the same speed as the dictation rate and also with 95% reliability, so that it does not interrupt the radiologist when dictating the report. The response time of the PACS function is not included as the system response time is based on the hardware and network setting. The model also illustrates examples of human agent response times which influence the RTT, such as the radiologist’s diagnosis time which should be conducted with consideration of the target RTT. The model shows consideration of the RTT in conducting the diagnosis task with the related three sub tasks.

Express the operational function step in cell (3,4):

This step expresses the operational perspective of functional models, the QR related to functional units and the communication among them. Although the goal and agent models above illustrate some functional units, this model explores further functional
aspects including input and output variables, related environment pre-and post-conditions and interrelations among functional units. Figure 52 illustrates the overall functional model of the radiology reporting process and related QR using a use case diagram. The overall QR of the VR report writing process are depicted through the QR notation associated with the system boundary; for example, the main QR goal of reducing RTT (QRID 1 time [RTT] <= target RTT). The QR related to functional units are depicted in the QR notation associated with use cases; for example, QRID 1.1 time [task access] within target RTT and QRID 3 patient-centred [worklist] are achieved with two use cases monitoring logs and scheduling the worklist where these two functions play a significant role in controlling the report writing time. The QR related to the communication between functions are depicted through the QR notation associated with the association links. For example, QRID 5 accessible [PMR] 24/7 is associated with the link that triggers access PMN.

![Figure 52 An Overall Operational Function for RRP Using Use Case Diagram](image)

A further detailed functional perspective can be also shown in this step. Figure 53 illustrates the input and output variables, the environment pre- and post-conditions and interrelations for the report dictation function and instant report typing function. In order to start report dictation the image has to be scanned and the diagnosis known. The output
of this function is the report diagnosis in voice format, which is the output of the VR typing function. The diagram also shows the QR associated with the VR typing function depicted through QR annotation.

Express the operational structure step in cell (3,3):

This step expresses construction of the structural model through outlining the system’s objects and relationships among them. Figure 54 groups all the objects found in the previous models with illustrations of the objects’ variables, functions and outlines of the associations among them with the related object QR. For example, the model shows worklist objects, variables and functions that influence QRID 1 time [RTT] by controlling QRID 1.1 time [tasks access] to be performed within target RTT. In doing so, this object is constrained by QRID 3 patient-centred [worklist] while scheduling the tasks in the worklist. The figure also introduces other objects related to system operations; for example, reporting backlog objects and verification backlog objects. These two objects are specialized worklists, one for reporting and the other for verification, which inherit the specification and QR details of the worklist object. The figure also depicts the QR related to system agents in their static status. For example, a radiologist needs to be 99.99% reliable to produce a reliable report, QRID 4.1 reliable [diagnosis ]>= 99.99%, as shown by QR annotation associated with the radiologist object.
Express the operational architecture step in cell (3,5):

This step expresses the construction of the architectural model of the operational concerns. This model expresses the distribution of the system components outlined in the operational models illustrated above with the related architectural QR. **Error! Reference source not found.** shows the radiology department agents inside the border and the external agents outside the boarder. For example, inside the radiology department RIS, PAS and archive are system agents; and radiologist and technician are human agents; and outside the radiology department physician is a human agent and HIS a system agent. The response time while communicating with system agents must be less than 5 seconds. For example, to achieve QRID 1.2.1 timely access [image] <5 seconds, the scanned images should be placed in the archive to reduce the load on the RIS. However, communication with human agents depends on agent availability, therefore the worklist plays a significant role in managing QRID 1.1 time [task access] within the target RTT with consideration of patient medical condition QRID 3 patient-centred [worklist]. The model also shows the QR related to the object communication among agents. For example, PACS should provide clear images to radiologist is depicted through QR annotation of QRID 6 clear [images] in
the association link between PACS and radiologist. The model also shows that immediate communication to update agent workload status influences other QR. For example, QRID 2.1 efficient [task access] can be achieved by immediately updating the worklist when a new scan is saved in PACS.

Express the operational behaviour step in cell (3,6):

This step expresses construction of the behavioural models through modelling the scenario of VR report typing and the state transition modelling of the report object. The sequence diagram illustrated in Figure 56 captures the VR report writing scenario with the related behavioural QR. It shows that the worklist is rescheduled when a new scan is saved or when a radiologist logs on to the RIS. Rescheduling the worklist according to these events influences achieving QRID 1.1 time [task access] within target RTT, and QRID 3 patient-centred [worklist]. Another example of the behavioural QR is QRID 2.1 efficient [task access] which is achieved through a sequence of three functions: retrieve images, retrieve PMR and send task notification with related data to the assigned radiologist. In this scenario a radiologist is informed of the next task with the related data instead of spending time enquiring about it. The state transition diagram in Figure 57 illustrates the actions and conditions to transform the status of the report object throughout the process, and the QR related to each object state. The model shows that the report diagnosis should have QRID 4 reliable [diagnosis report] >= 99.99% to avoid medical error, whereas the
report VR typing should have QRID 4.1 reliable [VR report writing] >= 95% to have a readable and understandable report.

Figure 56 Operational Behaviour for RRP Using Sequence Diagram

Figure 57 Operational Behaviour for Report Writing Object State Using Transition Diagram
8.5 Case study analysis and discussion

This section analyzes the above case study to demonstrate the framework's applicability in meeting the three objectives introduced in section 8.1.

The first objective is analyzing the completeness and traceability of QR models. The completeness is validated in two ways: (A) analyzing whether the QR models address all the components in the QR description form. (B) Analyzing whether the QR are expressed in different requirements modelling views.

The description form is used to define the QR captured at the early stage of the requirements process. This is demonstrated by a small table that addresses the QR description components as shown in section 8.4.1. At the later stage, the components of the QR description form are addressed in graphical models as shown in Figure 50. Furthermore, every QR annotated in the remaining models can be referred to as a child of the identified QR. Therefore, we can conclude that the QR description form is successful in expressing QR.

The case study demonstrates the data and models captured in each cell of the framework matrix with their related QR. The case study also demonstrates how the different perspectives capture and enrich the QR specifications. Although some QR have the same descriptions and annotations in several cells, each of them is addressed in the context of the perspective of the respective cell. The case study shows how every perspective differently enriches the QR specification in alignment with the development of the functional model. Therefore, we can conclude that the QR are expressed successfully in different requirements modelling views.

The traceability feature of the QRMF is validated through analyzing whether it is possible to trace upward and downward relationships among QR models following the sequence of QRIDs. It is possible to trace the QRID time [RTT] upwards (from operation to abstract level) and downwards (abstract to operation) in the SIG shown in Figure 50. Furthermore, all the other QR mentioned in the case study can be traced through their QRIDs. Therefore, we can conclude that the QR expressed in different perspectives are traceable.

The second objective is validated through analyzing whether the constructed QR models offer guidance. The case study expresses the QR through the QR description form and QR graphical annotations attached to the requirements models. Furthermore, the
QRIDs make it possible to trace the QR throughout the requirements modelling. These features enrich the conventional requirements models with a precise definition of QR. Therefore, we can conclude that the conventional requirements models are supported with related QR specifications.

Finally, the fulfilment of the third objective is a natural result of the positive evaluation criteria established above.

We acknowledge that the results from this case study can not necessarily be used as formal proof or generalize that the QRMF will always provide complete, traceable and comprehensible models of QR, for the following reasons:

- The case study was conducted in a research setting, which is different from the industrial setting.
- The models were developed by the author, who designed the framework.
- There is a lack of tools to support the framework models in an integrated way.

However, the case study adequately demonstrates the applicability, completeness, traceability and comprehensibility of the framework in the research setting, which is sufficient to demonstrate the framework's features for comparison against other QR modelling research.
9 Applicability and Comparison With Related Works

9.1 Introduction

The QRMF offers a generic solution to address QR in software systems, including a
generic structural schema supported by conceptual abstraction at four meta-modelling
levels and a generic modelling process. This chapter demonstrates the applicability of the
proposed QRMF and also compare its capabilities in relation to relevant work. The
applicability is demonstrated through demonstrating the generic and flexible features of
the framework's meta-model and modelling process. The meta-model, described in
Chapter 6, provides generic modelling concepts for the entirety of requirement modelling
views. Thus, it is sufficiently generic to flexibly represent the modelling concepts of a wide
range of requirements modelling notations. Section 9.2 demonstrates this through
evaluating the concepts of the meta-model against three modelling notations of two
requirement modelling views (goal- and architecture-oriented).

The framework modelling process introduced in Chapter 7 provides generic
guidelines to construct requirements models with their related QR without placing
restrictions on the modelling approach. Section 9.3 gives examples of adopting the
modelling process using two different requirement modelling approaches (object-oriented
and process-oriented modelling). Together with the case study in Chapter 6 (goal-oriented
modelling), a total of three examples of adopting the modelling process using three
different modelling approaches are used to demonstrate the applicability of the
framework process.

The demonstration is completed with an appraisal of the proposed framework in
terms of its capabilities and limitations. Section 9.4 presents an demonstration of the
QRMF features in comparison with related work.

9.2 Evaluation of the Applicability of the QRMF Meta-model

The framework is supported by a conceptual blueprint at meta-meta-level (Level 3)
to aid comprehension and the modelling process. This level of abstraction is generic, i.e. it
uses generic concepts of requirement modelling views and its flexibility enables its use
with various requirements modelling notations. Details of the meta-model can be found in
Chapter 6. This section demonstrates the generic and flexible features of the QRMF meta-
models by comparing the concepts conveyed with the meta-concepts of several modelling
notation examples. This demonstrates that the meta-model is rich enough to encompass
the modelling concepts of a wide range of modelling notations. The demonstration is conducted through six different examples of modelling notations, three from the goal-modelling view and three from the architectural modelling view.

9.2.1 Applicability to Convey Generic Concepts for Goal Modelling View

This section demonstrates the generic feature of the QRMF goal meta-model. The demonstration compares the concepts introduced in the meta-model with the meta-concepts related to three well known goal-oriented modelling notations (SD and SR from the i* framework, SIG from the NFR framework, and the goal model from KAOS) as shown in Table 12. The first column depicts the meta-concepts captured by the QRMF goal meta-model. The remaining columns show meta-concepts for the three modelling notations, with equivalent meta-concepts represented in the same rows.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal</td>
<td>Goal</td>
<td></td>
<td>Goal/Expectation/Requirement</td>
</tr>
<tr>
<td>Softgoal</td>
<td>Softgoal</td>
<td>Softgoal</td>
<td>Goal</td>
</tr>
<tr>
<td>Agent</td>
<td>Agent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Task</td>
<td>Operation softgoal</td>
<td></td>
</tr>
<tr>
<td>Object</td>
<td>Resource</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dependency link</td>
<td>Dependency link</td>
<td>Domain property</td>
<td></td>
</tr>
<tr>
<td>Refinement link</td>
<td>Refinement link</td>
<td>Refinement link</td>
<td>Refinement link</td>
</tr>
<tr>
<td>Contribution link</td>
<td>Contribution link</td>
<td>Contribution link</td>
<td>Obstacle</td>
</tr>
</tbody>
</table>

Table 12 shows that the SD and SR of the i* framework offer the richest goal modelling notations as they express more concepts related to the system and the domain. The table also shows that the QRMF goal meta-model represents all the meta-concepts found in the i* framework, although some of them have different names, e.g. object is equivalent to resource and function is equivalent to task. The SIG only conveys concepts related to the softgoals; other meta-concepts related to the goal and the domain are not included. The goal concept in KAOS is demonstrated in three different notations: goal, expectation and requirement. The goal can be behavioural or a softgoal, and both are represented with the same notation and treated alike. The expectation is that the goal is the responsibility of a single environment agent. The requirement is the goal under the responsibility of a single software agent. The dependency link in the QRMF goal meta-model is characterized by the domain properties, while KAOS expresses the same with a domain property meta-concept. The refinement meta-concept is the same for all the four
meta-concepts. Although the obstacle meta-concept is represented as a separate modelling view in KAOS [105], it reflects the negative contribution link in the QRMF goal meta-model. It can be concluded through this analysis that the QRMF goal meta-model is sufficiently rich to convey the modelling concepts of a wide range of goal modelling notations.

9.2.2 Applicability to Convey Generic Concepts for architecture Modelling View

This section demonstrates the generic features of the QRMF architecture meta-model. The demonstration compares the concepts introduced in the meta-model with the meta-concepts related to three well known modelling notations (BPMN, UML component diagram and UML communication diagram) as shown in Table 13. The first column depicts the meta-concepts captured by the QRMF architecture meta-model. The remaining columns show the meta-concepts for the three modelling notations; the equivalent meta-concepts are represented in the same rows.

### Table 13 Modelling Concepts of Architecture Modelling View

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>Swimlane (Pool &amp; lane)</td>
<td>System boarder</td>
<td></td>
</tr>
<tr>
<td>Agent</td>
<td>Swimlane</td>
<td>Actor</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Activity</td>
<td>Component</td>
<td></td>
</tr>
<tr>
<td>Sequence flow</td>
<td>Sequence flow and Gateway</td>
<td>Interface port Dependency</td>
<td>Sequence flow</td>
</tr>
<tr>
<td>Data flow</td>
<td>Object flow</td>
<td></td>
<td>Data flow</td>
</tr>
<tr>
<td>Object</td>
<td>message and object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td>Event</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QR</td>
<td>Artefact</td>
<td>Comment</td>
<td></td>
</tr>
</tbody>
</table>

Table 13 shows that the BPMN is the richest architecture modelling notation as it expresses more concepts related to the distribution of software agents and the communication among them. The table also shows that the QRMF architecture meta-model represents all the meta-concepts found in the BPMN apart from the artefact object. The artefact/comment object is used to add a comment to the architecture models for clarification, which is not a core concept in architecture modelling. Concepts in the same rows are equivalent but some of them have different names, e.g. architecture/location is equivalent to swimlane and function is equivalent to activity. The QRMF architecture meta-concept refers to the location of the objects, functions and agents in a software system. BPMN depicts the location with the swimlane meta-concept at two levels: pool for external agent and lane for internal agent. For this reason, swimlane is stated as
equivalent to the agent meta-concept. The system border demonstrates internal and external entities in the UML communication diagram. The sequence flow meta-concept is the only one found in all four examples, referring to how the modelling concepts communicate. The UML component diagram demonstrates the communication architecture among system functions whereas the UML communication diagram demonstrates the communication architecture among system agents. The BPMN and QRMF meta-concepts demonstrate the communication architecture among system agents and functions and also the data flow between them. Another meta-concept conveyed by QRMF and BPMN is the event meta-concept which characterizes the sequence flow and data flow of communication. The distinguishing feature of the QRMF meta-model is the only one of the architectural models to offer meta-concepts to express QR. It can be concluded through this analysis that the QRMF architecture meta-model is sufficiently rich to convey the modelling concepts of a wide range of architecture modelling notations.

9.3 Evaluation of the Applicability of the QRMF Modelling Process

The QRMF modelling process described in Chapter 7 helps construction of the requirements models with the related QR in a systematic, coherent and integrated way. The process offers an incremental method to identify, analyze and validate both FRs and NFRs/QR in an integrated multi-perspective modelling view. The process consists of three phases reflecting the development of the requirements knowledge and identifies steps within these phases to aid constructing the models related to the requirements modelling view in each row. The arrows between modelling activities, shown in Figure 32 in Chapter 7, do not imply a strict sequencing as many of the steps are intertwined or conducted in parallel, and backtracking may be required to check the consistency among models. Furthermore, this flexibility enables generic guidelines for adopting the QRMF with different requirement modelling approaches.

This section demonstrates the generic and flexibility features of the modelling process through evaluating the applicability of the modelling process to guide adoption of the QRMF framework through three different requirement modelling approaches: goal-oriented, object-oriented and process-oriented approaches. The example in Chapter 6 demonstrates adoption of the framework and the modelling process using the goal-oriented modelling approach. This section gives examples of using the framework and the modelling process for object-oriented and process-oriented modelling in the radiology reporting case study. The first phase of the modelling process is not included in the
demonstration because the strategic details are high-level and constructed from all the available resources without imposing a sequence on the steps related to a specific modelling approach. The second phase expresses the business requirements based on the details captured in the first phase and in further investigations of the domain. The third phase expresses the operational requirements and explains the technical details with consideration to the operation environment. This section demonstrated through examples of models of the last two phases. The demonstration is followed by analysis of the models to investigate if the modelling process is able to guide adoption of the framework without restricting the modelling approaches.

9.3.1 Applicability of the Modelling Process to Adopting the QRMF Via Object-Oriented Approach:

Object-Oriented Requirements Modelling (OORM) encapsulates information about the process, products and functionality of the domain into requirements objects, their inter-relationships and their reuse in different software settings. UML is the most popular object-oriented modelling notation, and is used in this evaluation. This section gives examples of the models captured in phases 2 and 3 using the object-oriented modelling approach.

Phase 2

Step 1: Express static model at the tactical level

![Figure 58 Static Model of Tactical Level in UML Class Diagram]
The static models at the tactical level express the concepts and relationships of business requirements in descriptive structural models. Figure 58 depicts these concepts and relationships in the radiology reporting example in a UML class diagram. The model is based on the information captured at the strategic level, shown in section 8.4.1 and from further analysis of the radiology reporting process. The objects in the diagram refer either to agent objects (radiologist, patient and physician) or to information objects (worklist and radiology information). The functional concepts are expressed within these objects, demonstrating the functions performed by each one. The goal concept is not demonstrated in the class diagram. The QR concepts included in this diagram are the QR that constrain an object, such as QRID 4 reliable [report], and the QR controlled by an object such as QRID 3 patient-centred [worklist] controlled by the worklist object. It can be observed that the model does not include software agents or the QR related to them.

**Phase 2**

**Step2: Express interactive model at the tactical level**

The interactive model of this phase expresses the architectural and behavioural views of business requirements. Figure 59 depicts the interactive model using a UML use case diagram, which expresses high-level architectural and behavioural views for business requirements through capturing the agents involved, the functions performed, the system
boundary and the association links that trigger them. This model complements the structure model shown in Figure 58 where the system agents (RIS, PACS and HIS), the functions performed by each of them and the related QR are depicted. The function concepts are depicted via use cases and the flow of firing those functions is depicted via association links. The QR captured in this model are the QR constraints of the functional units linked to the use cases and the generic QR goals of the overall system linked to system borders. It can be observed that the use case model captures additional QR to the ones captured in the static model and the strategic level, for example, QRID 1.2 time [diagnosis] within target RTT, QRID 1.2.1 time [consultation] within target RTT and QRID 1.3 time report writing = time report dictation. The sequence of QRIDs is allocated according to the flow of firing the functions.

Phase 3 Express the operational structure step in cell (3,3)

Figure 60 Object Model of Operational Level in Class Diagram

This step expresses construction of the operational structural model through comparing the system’s objects and relationships with the related object QR in the UML class diagram shown in Figure 60. The model is constructed through further analysis of the models and information captured in the previous phases with consideration given to the technical specifications of the system objects. The class diagram shown in Figure 60
has the same basic structure as the class diagram shown in Figure 58, but includes some specialisation of the objects such as the worklist object which is specialized into the reporting and verification worklists. It also includes more details to express objects’ properties and functions. The QR of this model include the object QR which refer to the QR constraining the objects and the QR controlled by an object. The QRID of the new QR expressed in this model is assigned after cross-checking the QR in the previous models before assigning the sequence QRID, because the class diagram does not include the sequence flow concept.

9.3.2 Applicability of the Modelling Process to the Adoption of the QRMF by the Process-Oriented Approach:

The business process modelling approach captures the flow of activities or services in conducting a business process. This approach captures the dynamic behaviour of a process that spans organizations, departments or agents. A process model is an interactive model commonly used as a communication tool and easily understood by stakeholders. Currently, the BPMN is commonly used in business process modelling and is therefore used in this evaluation.

Phase 2

Step1: Express static model at the tactical level

The process model is a behaviour type of model and is interactive; therefore, this step can not be conducted first with process-oriented. However, it is possible to capture the static concepts from an interactive model to construct the static model which reverses the order introduced in the modelling process at the tactical phase. This step is disregarded in this demonstration.

Step2 Express interactive model at the tactical level

Figure 61 depicts the interactive model of business requirements using a BPMN model. The diagram shows the main architectural and behavioural views for business requirements through the distribution of system agents and the communication flow among them. Each agent is represented by a lane; agents in a same department are grouped in a single pool. This demonstrates the external and internal distribution of agents’ locations. The function performed by each agent is demonstrated by an activity within the lane. The communication flow among those activities depicts the sequence flow and information flow. The QR captured in this model refer to the QR demanded of each of
these agents (expressed in the QR notation pointing to the agent swimlane); and the QR constrain the agents’ functions and the QR of interfaces that feature the communication between them. Although the model does not include the goal concept, all the QR/softgoals introduced in the previous phase are included using the introduced QR notations.

Figure 61 Interactive Model of Tactical Level in BPMN

Phase 3 Express the operational architecture step in cell (3,5):

Figure 62 Architecture Model of Operational Level in BPMN

This step expresses the operational architecture model with the related object QR in the BPMN model. The model outlines the system’s agents, their functions and the communication flow (sequence flow and data flow) among them as shown in
Figure 62. The model is constructed through functional decomposition and further analysis of the models and information captured in the previous phases with consideration given to the technical specifications of the software business process. The BPMN shown in Figure 62 has the same basic structure as the process model shown in Figure 61 but includes further details related to the system, such as introducing an archive as system agent to empower the QRID 1.1 response time of retrieving information (images and reports). Another example is introducing login and manage log activities to empower QRID 7 responsive worklist scheduling. The QR captured in this model also refer to the QR demanded of each agent, the QR constraining the activities and the QR of interfaces that feature the communication between them. The model in Figure 62 captures all the architectural QR shown in Figure 55 of Chapter 8 but with different QRIDs. For example, in the latter the QRID 1.21 refers to timely access [image] <= 5sec. which is expressed as QRID 1.1.1 timely access [image] <= 5sec.

9.3.3 Summary of the Analysis

The previous two sections provided examples of adopting the QRMF framework and modelling process using object-oriented and process-oriented modelling approaches. This section demonstrates the models constructed above, together with the models constructed the using goal-oriented modelling approach used in the case study in Chapter 8. The applicability of the modelling process is demonstrated through evaluating its generic features and flexibility in adopting the framework without restricting the modelling approach. The demonstration analyzes the models constructed by the three modelling approaches, focusing on whether the constructed models capture all the requirement modelling views, whether the constructed model captures a complete set of QR and whether it is possible to trace the captured QR through QRIDs. The table in Appendix B outline the requirements views and the QR captured in each of the models in the three approaches.

At the tactical level, phase2, the QRMF modelling process suggests capturing the entire requirement views with their related QR in a minimum number of models. At the goal-oriented modelling example in Chapter 8 captures the entirety of requirement views in three models: SD shown in Figure 46, SR shown in Figure 47 and BPMN shown in Figure 48. In addition to the QR elicited at the strategic level, these models capture further QR related to the refinement of goals in the SR model and related activities and their
communications in BPMN. The sequence of QRIDs is easily annotated following the refinement path in the SR model or the sequence flow in BPMN. At the object-oriented modelling example partly captures four modelling views in two models: the class diagram shown in Figure 58 and the use case diagram shown in Figure 59. Although these diagrams do not include the goal view, the QR demands/softgoals of the overall system are annotated by attaching the QR notation to the system border in the use case diagram. The QR of each object are annotated by attaching the QR notation in the class diagram and also the QR of each function are annotated by attaching the QR notation in the use case diagram. In addition to the QR elicited at the strategic level, this model captures further QR-related functions outlined in the use case. The sequence of QRIDs is annotated following the sequence flow between use cases. At the process-oriented modelling example captures all the views, excluding the goal view, in a single BPMN model, shown in Figure 61. The QR demands/softgoals of each agent are annotated by attaching the QR notation to each agent. In addition to the QR elicited at the strategic level this model captures further QR related activities and their communications. The sequence of QRIDs is annotated following the sequence flow in BPMN.

At the operational level, phase 3, a detailed model is proposed for each view, but for this demonstration only one example is given; therefore, the comparison is conducted in pairs with the object-oriented modelling proposed in the case study. The class diagrams in the goal-oriented modelling shown in Figure 54 in chapter 8 and the object-oriented modelling shown in Figure 60 capture the same modelling views. As in the structural model at the previous level, the QR of each object are annotated by attaching the QR notation in the class diagram. In addition to the QR elicited at the previous two levels, this model captures further QR related to new objects included in the diagram. The sequence of QRIDs is annotated following the hierarchical structure of the new objects; however, this requires cross-checking the sequence of QRIDs in the constructed models to ensure a traceable sequence is followed. In the other hand, the BPMN model in the process modelling shown in Figure 62 captures more views than the communication model in the goal-oriented model shown in Figure 55. As at the previous level, the QR demanded, the function QR and the communications QR are depicted for the detailed process model. The sequence of QRIDs also follows the flow of communication.

This shows that all three modelling approaches capture sufficient requirements modelling details with their related QR, whichever modelling approach is used. The demonstration shows that it is possible to establish complete QR by modelling the entire
views of requirements at each phase. The demonstration shows that it is easy to annotate the sequence of QRIDs following goal refinement paths and the object hierarchical structure in the static models; and the sequence flow in the interactive models. However, this is not entirely true for class diagrams, as the object hierarchical structure was insufficient to capture all the object QR, and therefore the QRIDs were annotated after cross-checking with the assigned QRIDs in the previous models.

9.4 Evaluation of QRMF against related work

Despite the importance given to addressing QR in the RE literature and the solutions proposed, in comparison with FRs, QR models are still immature and have not been widely adopted [3]. This research identifies two issues that contribute to this problem: the first is that the existing solutions do not fully support the critical characteristics related to the nature of QR and it is difficult to address those QR features without a structured and guiding framework. The second reason is that the existing frameworks that do offer comprehensive treatment of QR do not integrate QR with FR models; or, if they do, those solutions focus only on a single or limited view of requirements. It is difficult to include the entirety of FR views in a single model [24-26]. This issue is addressed by requirements modelling frameworks that integrate different modelling notations for different views of requirement, such as UML and KAOS. Although QR are inherently more difficult to address than FRs [19, 23], the RE literature does not offer comprehensive support for QR in different requirements modelling views.

To address these limitations, this research proposed the QRMF with consideration given to the challenging features of QR and the limitations of existing solutions. The framework does not disregard the strength of existing solutions; however, it offers a structured platform on which to employ those solutions with consideration given to QR. This section compares the framework’s features with the features of work related to the process-oriented approach to QR modelling and approaches that extend requirements models with QR.

9.4.1 Process-Oriented Approaches to QR

To the best of our knowledge, the NFR framework [19] offers the most comprehensive treatment of QR. It is provides a constructive mechanism for eliciting, analyzing, interrelating and determining the impact of decisions upon QR. It records these actions in a graphical representation using a SIG. This notation enables building customized solutions with consideration to the characteristics of the particular domain.
Furthermore, the NFR framework facilitates traceability among the identified QR through refinement links captured in the SIG model. Although the NFR framework captures the QR from high-level softgoals to the operational options, and also aids selection decisions, it only considers the goal view of requirements in a static model of the software system. This does not integrate the defined QR with FRs models.

The i* framework [14] offers a strategic modelling framework for RE. This model explains why a new system is needed, identifies the new system configurations and processes, and demonstrates the extent to which the FRs and NFRs meet users’ needs. The framework facilitates eliciting QR as softgoals with the related FRs and environment features at an early stage of requirements. It also facilitates QR refinement into lower levels until operational options are reached. As NFRs, the i* also facilitates QR analysis and determines the impact of decisions upon QR. Although the framework expresses QR in integration with FRs and the domain context, it offers only a static view of requirements.

KAOS [69] is a goal-oriented framework that offers a multi-modelling paradigm (graphical and formal) of different requirement views. It is a methodology for eliciting, specifying and analyzing goals, requirements, scenarios and responsibility assignments. The formal modelling supports various forms of formal reasoning through real-time temporal logic. QR are treated as goals and are therefore modelled and analyzed with the same notation. Lamsweerde [80] proposed a meta-model and modelling process to express requirements using KAOS. The KAOS framework is the closest approach to the QRMF, as both offer multi-perspective models, guided modelling process and meta-models. However, in QRMF multi-perspective modelling has two dimensions: the requirements modelling view and the progression of requirements knowledge. QRMF does not offer a new modelling tool but proposes a generic approach to be adopted using existing modelling languages. The QRMF graphical annotations to QR in requirements models provides an equivalent emphasis for both FRs and QR modelling.

The Performance Requirements Framework (PeRF) [23] offers a systematic approach to managing performance requirements. It catalogues knowledge about performance concepts, software performance engineering principles and information systems development, and represent them using the NFR Framework. This process-oriented approach to QR allows customized solutions to be built, taking into account the characteristics of the particular domain. Although the framework considers software development stages, its supports for QR is limited to the goal-oriented view of
requirements. In contrast, the QRMF expresses QR through the entire range of views of requirements, and focuses only on the RE stage.

The WinWin paradigm of QR [166] aims to identify and resolve conflicts among stakeholders’ conflicting requirements. This approach is based on a spiral model for requirements elicitation, identifying and resolving requirements conflicts by drafting and negotiating artefacts such as win conditions, issues, options and agreements. This approach is also supported with a knowledge-based tool to enhance process effectiveness and efficiency requirements elicitation. This paradigm and the supporting tool focus on QR elicitation with related satisfying measures using natural language. This tool can support QRMF in eliciting the strategic requirements as it do not involve requirement modelling.

Component-Bus-System-Property (CBSP) [167, 168] is a lightweight approach that aims to provide a systematic way of reconciling requirements and architectures. It leverages architectural concepts to recast the requirements to be mapped to architectures. It is a process-oriented approach consisting of five steps starting from requirements elicitation and finishing with selection decisions of the architectural elements. The approach is based of a taxonomy of architectural dimensions. The input of the CBSP method is a set of general requirements in textual form, and the output is an intermediate model that captures architectural decisions for software systems. CBSP facilitates capturing and maintaining complex relationships between requirements and architectural components; however, it isolates QR in requirements at the system level and architectural element level. The resulting model consists of several levels of requirements, from generic requirements down to operational ones. This covers the vertical dimension of the QRMF and only the goal view of requirements.

Tsadimas et al. [169] proposed a model-based approach that emphasizes NFRs in an Enterprise Information System (EIS) architecture. Three system design views are adopted (functional, topology and network), each of which addresses a different kind of requirement. The NFRs are integrated into the EIS architecture model where they are satisfied by specific entities included in the three system design views. A meta-model is proposed to describe the relationships between NFRs and the architectural components. SysML is adopted as a modelling language as it enables requirements definition and can be formally extended. Although this work offers some guidelines on what is required to model NFRs (consistency, relationships and notation extension), only goal, agent, function
and architecture requirement views are included at a the operational level of the QRMF schema.

The QRF (Quality Requirements of a software Family) [170] method offers a systematic approach that focuses on how to define, represent, trace and transform QR into architectural models. It aims to enable quality demonstration in the early phases of software product family development, conducted through five steps: impact analysis, quality analysis, variability analysis, hierarchical domain analysis and quality representations. Impact analysis defines the interested stakeholders and the goals concerning the quality of the family. Quality analysis separates the quality concerns related to business, constraints and functionality. Variability analysis identifies any variability in quality. The hierarchical domain analysis brings together the required information for modelling and evaluating the architecture, and quality representation represents QR in architectural models. These steps are undertaken at three levels of requirements development: business, domain and product family, and they are equivalent to the rows of the QRMF representational schema (strategic, tactical and operational). Unlike QRMF, QRF annotates QR only in the architectural view of requirements models through UML profile extension.

In addition to QRF, the RE literature proposes many other QR-aware methods for designing product-line software architectures, such as Quality-driven Architecture Design and quality Analysis (QADA) [171] and QUALity driven System ARchitecting (QUASAR) [172]. These approaches are limited to the architectural view of requirements and are also specific to software the product line approach to software development.

9.4.2 Extension of Requirements Models to Address QR

It is not possible to express meaningful QR without relating them to a functional concept [18]; otherwise, the QR are generic and out of context. Some authors therefore recommend that QR should always be linked to FRs [173]. However, QR have been modelled in isolation from FRs. For example, the NFR framework [19] offers advanced treatment of QR but does not stress conflict detection with FRs. Another example is ISO 25000 which offers a product-oriented approach to QR through using the QR model highlighted in ISO 9126 [45]; this model is generic and is difficult to configure with the functional model of a specific software project. One of QRMF's features is to annotate requirements models with QR. This concept is not new and is widely accepted in RE research. The first to introduce it were Cysneiros et al. [173], followed by a wide range of
approaches and tools introduced to support this integration in different models. This section illustrates some of the proposed approaches that focus on two common modelling approaches: the object-oriented modelling using UML [174] and business process modelling using BPMN [175].

### 9.4.2.1 Expressing QR in UML

Cysneiros et al. [173] proposed the adoption of a goal-oriented method, with supporting representations, to be used in conjunction with object-oriented models; i.e. integrating NFRs into object models. Their work used the notion of a lexicon, the Language Extended Lexicon (LEL), as an anchor to integrate NFRs into object models. The purpose of LEL is to register project vocabularies (FR and NFR symbols) and determine the relationship between them. The relationship between the NFR representation schema and the conceptual object model is the entries in the application lexicon. These links are used as heuristics in order to guide the software engineer in the early integration of NFRs into the requirements model. This proposal supports integrating NFRs only into UML class diagrams. The LEL concept is not widely accepted in the software industry as it significantly increases the workload of systems analysts [174].

Krishna and Gregoriades proposed an extension to UML diagrams with QR [174] through comprehensive and traceable declaration of QR and also a modelling process that facilitates possible integration of QR in different UML diagrams. This approach captures QR in different views using UML notation. Their work has inspired this research as it stresses the importance of comprehensive and traceable declarations of QR. QRMF is generic and can be adopted without imposing specific modelling notations.

Supakkul and Chung [27] proposed a framework to represents NFRs with FRs in UML use case diagrams through four integration points: system boundary, actor, use case, and communicate association. NFRs are represented as softgoals and associated with appropriate UML use case diagram notation. The framework is supported by NFR propagation rules and a meta-model. The NFR propagation rules are proposed to eliminate redundant NFR specifications. The meta-models demonstrate the integration semantics of NFR with the UML meta-model using a UML profile [176]. This work also drew the attention of this researcher to the importance of relating different types of NFR to different types of modelling notation. However, the proposal is limited to a UML use case diagram, while QRMF offers a generic approach to modelling QR without restricting modelling notation.
Abid et al. [177] extended a UML profile to include goal-oriented modelling which comprises QR as softgoals. The profile extension is tested and implemented in the jUCMNav Eclipse plug-in. It did not include QR annotation at the modelling level or guidelines to employ this extended profile.

Brito et al. [178] presented a process-oriented approach to identify, specify and integrate QR with FRs. Their approach consists of three main activities: identification, specification and integration of requirements. The first activity identifies FRs and QR related to the domain and stakeholders. Then FRs are specified through UML models and QR are specified using the introduced template of quality attributes. This activity also identifies the cross-cutting QR attributes. The final activity integrates QR with FRs through annotating QR as UML use cases and sequence diagrams. The three activities of this approach refer to requirement knowledge development; however, these three activities do not reflect the same depth offered by QRMF and only address QR through two views of requirements. Both the QR template and the QR description form of QRMF emphasize the importance of explicit description of all QR details.

9.4.2.2   Expressing QR in Business Process Models

The BPMN provides guidelines to support notation extensions with additional modelling features [179]. A wide range of research targets extending BPMN with QR, such as extending BPMN with business goals and performance measures [180], cost and reliability requirements extension [181] [182], security requirements extension [183] and time extension [184].

Korherr and List extended BPMN, Event-Driven Process Chain (EPC) [180] and UML 2 activity diagrams [185] to model goals and performance measures by annotating cost, time and other quality values. Their quality measures are expressed by customer satisfaction rate or customer complaint rate. This work also proposed a meta-model extension for the additional elements in BPMN and EPC; and UML profile extensions for activity diagrams. QR are expressed in these three modelling notations through labels representing QR values in a process model [180]. This work focuses only on the process-oriented approach to requirements. The process model captures only architectural, functional and to some extent agent views of requirements.

Magnani and Montesi extended BPMN notation with a cost annotation which represents cost value, cost interval, or average cost [181]. They also proposed an overall business process cost evaluation that considers the heterogeneous behaviour of a business
process [181]. Cost and reliability values were also represented by textual annotations in [182]. Path and Wiring [182] proposed four patterns for QR and business process model evaluations: sequential, parallel, conditional and successive possibilities. These two proposals capture only architectural and functional views of requirements.

Kawther et al. [9] extended BPMN with customer-facing QR. The extension is introduced into the BPMN meta-model and also into the BPMN modelling notation. This work also proposes an analytical based evaluation model, using reduction rules, for evaluating overall business process quality requirements. It addresses QR at the business process level, which is only related to the functional view of requirements. The QR are described with their related measures, but this definition can be subjective as it does not capture sufficient details to comprehensively identify and trace QR.

Gagné and Trudel proposed Time-BPMN which extends BPMN with attributes and properties to capture the temporal constraints and dependencies of a business process which have visual depictions and control activity workflow in a business process [184]. Time-BPMN enhances the workflow description from a business process functional view of requirements.

9.5 Summary
The framework features was demonstrated through two mechanisms: empirical and theoretical evaluation. The empirical demonstration investigated the ability of the framework to offer the features introduced. It was conducted through the case study of a radiology reporting process, detailed in the previous chapter. The first part of this chapter also offered an empirical demonstration of the generic features of the framework. Two types of applicability demonstration were introduced using examples from the case study. The first demonstrated the generic features of the QRMF meta-model to include modelling concepts related to all the requirement modelling views without restricting modelling notations. The demonstration was conducted through six modelling notation examples, three of the goal view and three of the architecture view. The result of this investigation showed that the QRMF meta-model is sufficiently generic to include the modelling concepts of all the requirement modelling notations. The second empirical demonstration investigated the generic features of the QRMF modelling process to be adopted without restricting the modelling approach. Section 9.3 gave two examples of adopting the modelling process using object-oriented and process-oriented modelling. Together with the example in Chapter 8 which employed goal-oriented modelling, a total of three
examples were demonstrated for this evaluation. The result of this investigation showed that the QRMF modelling process is generic and can guide the modelling process without restricting the approach.

The chapter also provided theoretical demonstration mechanisms through a comparative appraisal of the QRMF with different approaches treating QR. These approaches were reviewed in two main groups: the process-oriented approach to QR and approaches that extend well known modelling notations with QR such as UML and business process modelling notation. QRMF gathered features of these approaches and offers a platform to use some of them in a constructive and guided manner. The result of this comparison shows that all the QRMF features are accepted concepts in the RE literature, and none of the above related work proposed all the features offered by QRMF in a single framework.
10 Conclusion

QRMF offers a process-oriented approach to constructing a fairly complete QR models for a given project. The QR models are constructed through a QR description form, a multi-perspective QR modelling schema, generic meta-model, and a guiding modelling process. The QR description form includes QR elements related to identifying, analyzing and validating QR of a software product in a traceable and unambiguous fashion. The multi-perspective QR models are expressed in a descriptive representation schema that cohesively fits the entirety of requirement modelling views. The structural representation schema plays a significant role in guiding the construction of requirements models comprehensively and with consistency. The meta-model is significant in capturing the framework features in a comprehensive and generic fashion. The modelling process facilitates constructing requirements models with their related QR in a systematic, coherent and integrated way. The framework is generic and can be adopted without restricting the modelling notation or modelling approaches.

The QRMF features was demonstrated empirically and theoretically. The empirical testing was conducted through the case study of a radiology reporting system to demonstrate the utility and applicability of the framework's features. The theoretical appraisal was conducted through comparison with related work.

This chapter outlines the work reported in this thesis, highlighting the most important findings and contributions. It offers conclusions about the benefits and limitations of the research. Finally, it presents suggestions for future work.

10.1 Thesis Contributions

The aim of this thesis is to offer a QRMF that can guide the construction of complete and comprehensive QR models through adopting a generic, structured and guided multi-perspective modelling framework. This framework consists of four novel components: a QR description form, a multi-perspective representation schema, a generic meta-model, and a generic guiding modelling process. These contributions are summarized as follows:

**Contribution 1: QR description form**

The QR description form is a template that aids identification of QR in a complete, traceable and unambiguous fashion. The description form consists of elements related to
identifying, analysing and validating QR. These elements are appropriate for both early and late requirements phases.

The QR description form identifies QR with an QRID, which introduced in this research. The QRIDs flag each QR with an ID that can be traced throughout requirement development and also help to match the fulfilment of the QR against the owner's expectations.

**Contribution 2: QR description form supporting schema**

The elements of the QR description form and the relationships between them are captured in a QR structural schema. The schema is presented in an Relational schemawhich can easily be adopted by a database management system or XML documents to develop a tool for modelling, analyzing and validating QR.

**Contribution 3: A multi-perspective representation schema**

The QRMF offers a multi-perspective modelling framework to address QR concepts in a structured way, captured in the QRMF representation schema. The schema is a matrix of three rows and six columns. The rows represent the transformation of requirements from business to technical levels through strategic, tactical and operational levels. The columns represent all RE modelling views: goal, agent, structure, function, architecture and behaviour. The intersection of each row and column is a cell with a unique perspective of requirements and related QR. Alternatively, when cells are grouped together a complete requirements model is formed, reflecting the development of requirements with their related QR. This structure helps in acquiring the necessary knowledge systematically and cohesively.

**Contribution 4: Generic meta-model**

The QRMF meta-model is a high-level abstraction that helps understanding of the framework’s features, further guides the modelling process and is also a reference blueprint for developing a modelling tool applicable to the framework. The meta-model is generic, i.e. it is able to delineate generic concepts related to different modelling views independently of specific modelling notations or approaches. The meta-model is represented in a UML class diagram, which is widely known among requirements engineers and can be used as a modelling template to guide the generation of models for each cell in the framework.
Contribution 5: Generic modelling process

The QRMF modelling process offers a constructive method for building an entire requirements model out of the framework perspectives. The framework modelling process consists of three phases that reflect the development of the requirements knowledge, represented in the framework matrix rows. At each phase a method elaborates the process of identifying and expressing the concepts related to each cell in the row. The flow among the cells indicates the interrelation among the modelling perspectives. The arrows do not refer to a strict sequence, as many of the steps are intertwined, conducted in parallel, and may require backtracking to check consistency among the models. The modelling process offers generic guidelines on the use of the framework by adapting different requirement modelling approaches.

Contribution 6: Example of using the QRMF in a case study

An example of how the QRMF can be applied in an industrial project is given through the case study of a radiology reporting system. The case study demonstrates the utility of the framework using a goal oriented approach and also used to demonstrate the applicability of using the framework with different modelling notations and modelling approaches.

10.2 Limitations of the Research

The framework's limitations can be summarized as follows:

- The traceability of the identified QR is through the sequence of QRIDs. However, the framework does not offer graphical means to depict the links in tracing this sequence at different diagrams as it is offered by the NFR framework [6] and Krishna and Gregoriades’ UML annotation with QR [174]. Graphical traceability of QR makes it easier to check the consistency and fulfilment of QR’ evolution to meet stakeholder requirements.

- The framework’s empirical demonstration in the case study, and the applicability of the meta-model and modelling process were conducted in a research environment. Although the results of this demonstration verify the framework's features, applying them same in an industrial case study would introduce new variables which might introduce new challenges.
• The framework’s meta-models and QR description schema were designed using the notations of UML class diagrams and ER diagrams, respectively. The proposed schemas were not formally evaluated or designed with a modelling tool specifically intended for this purpose.

10.3 Future work

There are many possible research directions for the new multi-perspectives modelling approach to QR. These can be broadly grouped into categories: tool support and evaluation in an industrial setting.

**Tool support for QRMF:** the implementation and validation of a tool to integrate modelling of QR with FRs. Such a tool is important for this framework to facilitate automatic traceability and consistency checking among requirements models. The tool should support the following features:

• QR description schema: to facilitate a comprehensive traceable definition of QR and also limit the redundancy among QRs definitions.
• QRMF meta-models: to include all the meta-concepts and meta-relationships among those concepts to guide the modelling process and model validation.
• A library of modelling notations with a QR extension to allow the requirements engineer to use the framework without restricting the modelling approach or notation.
• A parser to capture strategic FRs and NFRs forms and documentation related to the project.

**Empirical evaluation:** the case study demonstration in an industrial setting has been postponed until the modelling tool has been developed, because it would be difficult to construct the multi-perspectives models that integrate QR with FRs models without a supporting tool. This work is important for the framework validation.
References


196


## Appendix A

List of 227 distinct QR found in the literature with the frequency found in the commonly known 10 QR models

<table>
<thead>
<tr>
<th>Quality Requirement</th>
<th>Quality Requirement</th>
<th>Quality Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access audit</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Access control</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Accessibility</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Accountability</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Accuracy</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Adaptability</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Additivity</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Adjustability</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Aesthetics</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Affordability</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Analyzability</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Assurance</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Attractiveness</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Augmentability</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Availability</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Buffer Space Performance</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Capability</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Capacity</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Changeability</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Clarity</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Code Space Performance</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Co-existence</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Cohesiveness</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Conformance</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Commonality</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Communication</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Communication commonality</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Communication Cost</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Communication time</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Communicativeness</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Compatibility</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Competence</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Compatibility</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Commonality</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Communication</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Communication Cost</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Communication time</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Communicativeness</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Compatibility</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Competence</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Commonality</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Communication</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Communication Cost</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Communication time</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Communicativeness</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Compatibility</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Competence</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Quality Requirement</td>
<td>Quality Requirement</td>
<td>Quality Requirement</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Completeness</td>
<td>3</td>
<td>Space performance</td>
</tr>
<tr>
<td>Main memory performance</td>
<td>1</td>
<td>Space performance</td>
</tr>
<tr>
<td>Component integration</td>
<td>1 Maintainability</td>
<td>Specificity</td>
</tr>
<tr>
<td>Cost</td>
<td>1</td>
<td>Speed</td>
</tr>
<tr>
<td>Composability</td>
<td>1 Maintenance cost</td>
<td>Stability</td>
</tr>
<tr>
<td>Comprehensibility</td>
<td>1 Maintenance time</td>
<td>Standardisability</td>
</tr>
<tr>
<td>Conceptuality</td>
<td>1 Maturity</td>
<td>Start up time</td>
</tr>
<tr>
<td>Conciseness</td>
<td>2 Mean performance</td>
<td>Storage efficiency</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>1 Measurability</td>
<td>Structuredness</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>1 Memory utilisation</td>
<td>Subjevicy</td>
</tr>
<tr>
<td>Configurability</td>
<td>1 Mobility</td>
<td>Supportability</td>
</tr>
<tr>
<td>Consistency</td>
<td>5 Modifiability</td>
<td>Surety</td>
</tr>
<tr>
<td>Controllability</td>
<td>1 Modularity</td>
<td>Survivability</td>
</tr>
<tr>
<td>Coordination Cost</td>
<td>1 Naturalness</td>
<td>Susceptibility</td>
</tr>
<tr>
<td>Coordination Time</td>
<td>1 Observeability</td>
<td>Sustainability</td>
</tr>
<tr>
<td>Correctness</td>
<td>4 Off-peak-period performance</td>
<td>Suitability</td>
</tr>
<tr>
<td>Cost</td>
<td>2 On-time delivery</td>
<td>Tangibility</td>
</tr>
<tr>
<td>Coupling</td>
<td>1 Operability</td>
<td>Testability</td>
</tr>
<tr>
<td>Customer evaluation time</td>
<td>1 Operating cost</td>
<td>Testing time</td>
</tr>
<tr>
<td>Customer Loyalty</td>
<td>1 Peak-period performance</td>
<td>Throughput</td>
</tr>
<tr>
<td>Customer relationship</td>
<td>1 Performance</td>
<td>Time</td>
</tr>
<tr>
<td>Customisability</td>
<td>1 Performance</td>
<td>Time behaviour</td>
</tr>
<tr>
<td>Data analysis time</td>
<td>1 Planning cost</td>
<td>Time performance</td>
</tr>
<tr>
<td>Data commonality</td>
<td>1 Planning time</td>
<td>Timeliness</td>
</tr>
<tr>
<td>Data Space performance</td>
<td>1 Plasticity</td>
<td>Tolerance</td>
</tr>
<tr>
<td>Decomposability</td>
<td>1 Portability</td>
<td>Traceability</td>
</tr>
<tr>
<td>Degradation of service</td>
<td>1 Portability compliance</td>
<td>Trainability</td>
</tr>
<tr>
<td>Dependability</td>
<td>2 precision</td>
<td>Transferability</td>
</tr>
<tr>
<td>Development cost</td>
<td>2 Predictability</td>
<td>Transparency</td>
</tr>
<tr>
<td>Development time</td>
<td>2 Privacy</td>
<td>Understandability</td>
</tr>
<tr>
<td>Device efficiency</td>
<td>1 Process management time</td>
<td>Uniform performance</td>
</tr>
<tr>
<td>Disaster recovery</td>
<td>1 Processing time</td>
<td>Uniformity</td>
</tr>
<tr>
<td>Distributivity</td>
<td>1 Production life span</td>
<td>Usability</td>
</tr>
<tr>
<td>Diversity</td>
<td>1 Productivity</td>
<td>Usability compliance</td>
</tr>
<tr>
<td>Documentation</td>
<td>1 Project stability</td>
<td>User-friendliness</td>
</tr>
<tr>
<td>Domain analysis cost</td>
<td>1 Project tracking cost</td>
<td>Utility</td>
</tr>
<tr>
<td>Domain analysis time</td>
<td>1 Promptness</td>
<td>Validity</td>
</tr>
</tbody>
</table>

202
<table>
<thead>
<tr>
<th>Quality Requirement</th>
<th>Quality Requirement</th>
<th>Quality Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficiency compliance</td>
<td>Prototyping time</td>
<td>Workload</td>
</tr>
<tr>
<td>Empathy</td>
<td>Recovery time</td>
<td>Wrappability</td>
</tr>
<tr>
<td>Enhanceability</td>
<td>Reengineering cost</td>
<td></td>
</tr>
<tr>
<td>Effectiveness</td>
<td>Reconfigureability</td>
<td>Wrappability</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Recoverability</td>
<td>Visibility</td>
</tr>
<tr>
<td>Elasticity</td>
<td>Recovery</td>
<td>Workload</td>
</tr>
</tbody>
</table>
## Appendix B

Captured QR using different requirement engineering approaches

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>QRID 1</td>
<td>QRID 1</td>
<td>QRID 4</td>
<td>QRID 1</td>
<td>QRID 8</td>
<td></td>
</tr>
<tr>
<td>QRID 2</td>
<td>QRID 2</td>
<td>QRID 4.1</td>
<td>QRID 2</td>
<td>QRID 9</td>
<td></td>
</tr>
<tr>
<td>QRID 3</td>
<td>QRID 3</td>
<td>QRID 4.2</td>
<td>QRID 3</td>
<td>QRID 8</td>
<td></td>
</tr>
<tr>
<td>QRID 4</td>
<td>QRID 4</td>
<td>QRID 4.3</td>
<td>QRID 4</td>
<td>QRID 7</td>
<td></td>
</tr>
</tbody>
</table>

**Goal Oriented**

- **Phase 2**
  - QRID 1
  - QRID 2
  - QRID 2.1
  - QRID 3
  - QRID 4
  - QRID 4.1
  - QRID 4.2
  - QRID 4.3
  - QRID 5
  - QRID 6
  - QRID 7
  - QRID 8

- **Phase 3**
  - QRID 1
  - QRID 1.1
  - QRID 1.2
  - QRID 1.2.1
  - QRID 1.2.2
  - QRID 1.2.3
  - QRID 1.3
  - QRID 2
  - QRID 2.1
  - QRID 3
  - QRID 4
  - QRID 4.1
  - QRID 4.2
  - QRID 5
  - QRID 6
  - QRID 7
  - QRID 8
  - QRID 9

**Object Oriented**

- **Phase 2**
  - QRID 1
  - QRID 2
  - QRID 3
  - QRID 4
  - QRID 4.1
  - QRID 5
  - QRID 6
  - QRID 7

- **Phase 3**
  - QRID 1
  - QRID 1.1
  - QRID 1.2
  - QRID 1.2.3
  - QRID 2
  - QRID 3
  - QRID 4
  - QRID 4.1
  - QRID 5
  - QRID 6
  - QRID 7
  - QRID 8
  - QRID 9

**Process Oriented**

- **Phase 2**
  - QRID 1
  - QRID 2
  - QRID 2.1
  - QRID 3
  - QRID 4
  - QRID 4.1
  - QRID 4.2
  - QRID 4.3
  - QRID 5
  - QRID 6
  - QRID 7
  - QRID 8
  - QRID 9

204
<table>
<thead>
<tr>
<th>Phase3</th>
<th>QRID 1</th>
<th>QRID 1.1</th>
<th>QRID 1.1.1</th>
<th>QRID 1.1.2</th>
<th>QRID 1.1.3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>QRID 1.2</td>
<td>QRID 1.3</td>
<td>QRID 1.2.1</td>
<td>QRID 2</td>
<td>QRID 2.1</td>
</tr>
<tr>
<td></td>
<td>QRID 5</td>
<td>QRID 6</td>
<td>QRID 7</td>
<td>QRID 8</td>
<td>QRID 9</td>
</tr>
</tbody>
</table>