Cross Functional Organisational Embedded System Development

Sophie Lennon
Masters by Research 2015
Supervised by Prof. Ita Richardson

Submitted to the UNIVERSITY of LIMERICK, November 2015
Declaration

The work presented in this thesis is entirely my own work. It has not been submitted previously to this or any other institute for this or any other academic award. Where use has been made of the work of other people, it has been acknowledged and referenced.

Signed: _____________________ Date: ________________

Sophie Lennon
I would like to thank my supervisor Prof. Ita Richardson, an incredibly vibrant, encouraging and colourful person. Thank you!

My colleagues, who were very generous in sharing their time and knowledge, after they finished questioning me on why I wanted to give myself more work.

This research has been funded by Science Foundation Ireland through IRCSET under Grant no. EGBP/2013/54 and supported by Lero. I would like to thank them for the opportunity.
“Studying about that good ‘ol way”
Embedded system development is continuing to grow. Medical, automotive and Internet of Things are just some of the market segments. There is a tight coupling between hardware and software when developing an embedded system, often needing to meet strict performance targets, standards requirements and aggressive schedules. Embedded software developers need to consider hardware requirements in far greater detail as they can have a significant impact on the quality and value of the software to be delivered.

Metrics and standards are widely used when evaluating software quality. Embedded software should continue to be measured against traditional software quality criteria derived from standards like ISO 9000 and CMM/CMMI. However in order for software to add value to an embedded system, key factors from other functional areas should also be incorporated. There is no model, which demonstrates the unique factors that impact embedded software. Software can be impacted either directly or indirectly either directly through the software development process or indirectly via the supporting organisation and divisions. The model I have proposed identifies software quality criteria that is unique to embedded software quality and outlines significant areas and factors that impact the success of developing embedded software.
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/D Conversion</td>
<td>Analog to Digital Conversion</td>
</tr>
<tr>
<td>API</td>
<td>Application Program Interface</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application Specific Integrated Circuit</td>
</tr>
<tr>
<td>CMM</td>
<td>Capability Maturity Model</td>
</tr>
<tr>
<td>CMMI</td>
<td>Capability Maturity Model Integration</td>
</tr>
<tr>
<td>CSF</td>
<td>Critical Success Factors</td>
</tr>
<tr>
<td>D/A Conversion</td>
<td>Digital to Analog Conversion</td>
</tr>
<tr>
<td>ESQM</td>
<td>Embedded Software Quality Model</td>
</tr>
<tr>
<td>ETSI</td>
<td>European Telecommunications Standards Institute</td>
</tr>
<tr>
<td>FAA</td>
<td>Federal Aviation Administration</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>GSD</td>
<td>Global Software Development</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electronic Engineers</td>
</tr>
<tr>
<td>IOT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>ISO</td>
<td>International Standards Organisation</td>
</tr>
<tr>
<td>KPA</td>
<td>Key Practice Areas</td>
</tr>
<tr>
<td>KPI</td>
<td>Key Performance Indicator</td>
</tr>
<tr>
<td>SLOC</td>
<td>Source Lines of Code</td>
</tr>
<tr>
<td>M&amp;A</td>
<td>Mergers and Acquisitions</td>
</tr>
<tr>
<td>MNC</td>
<td>Multinational Company</td>
</tr>
<tr>
<td>NSAI</td>
<td>National Standards Authority of Ireland</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>Acronym</td>
<td>Definition</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>ROI</td>
<td>Return on Investment</td>
</tr>
<tr>
<td>RTL</td>
<td>Register Transfer Language</td>
</tr>
</tbody>
</table>
Chapter 1 - Introduction ............................................................................. 1
  1.1 Research Objective........................................................................... 1
  1.2 Background ...................................................................................... 1
  1.3 Motivation......................................................................................... 2
  1.4 Summary of Research ...................................................................... 2
  1.5 Ensuring Quality & Value................................................................. 3
  1.6 Limitations of the Research ............................................................. 4
  1.7 Research Bias.................................................................................. 4
  1.8 Structure of the Thesis ................................................................. 5

Chapter 2 – Literature Review ................................................................... 6
  2.1 Introduction........................................................................................ 6
  2.2 Literature Review Structure ............................................................. 6

  Section 1 Embedded Systems ............................................................... 8
  2.3.1 Embedded Systems ..................................................................... 8
     2.3.1.1 Growth in Embedded Systems ........................................... 9
     2.3.1.2 Embedded Systems Design ............................................. 9
     2.3.1.3 Embedded Systems Process .......................................... 10
  2.3.2 Role of Software Engineering in Embedded Systems Development
     ........................................................................................................... 10
  2.3.3 Embedded Software Complexities .......................................... 11
  2.3.4 Challenges of the Embedded Software Development Process to
     Software & Hardware Developers ..................................................... 13
  2.3.5 Complexities in Embedded Systems Development ................. 13

  Section 2 - Development Stress ............................................................. 15
2.4.1 Project Success

2.4.1.1 Project Success Criteria

2.4.2 What is Development Stress?

2.4.2.1 Managing Development Stress

2.4.2.2 Development Stress Model

2.4.2.3 Mitigating Development Stress

2.4.3 Inputs to the ESQM-V1

Section 3 - Global Software Development

2.5.1 Why Global Software Teams?

2.5.1.1 Challenges of Global Software Development

2.5.1.2 Communication and Coordination

2.5.1.3 Team Distribution

2.5.2 Mergers and Acquisitions in GSD

2.5.2.1 Merger and Acquisition Integration Stages

2.5.3 Complexities in GSD Communication

2.5.4 Difficulties in M&A and GSD

Section 4 - Software Quality

2.6.1 Quality

2.6.1.1 Quality Assurance

2.6.1.2 Quality Standards

2.6.2 Software Quality

2.6.2.1 Software Quality Assurance

2.6.2.2 Software Quality Requirements for Embedded Systems

2.6.2.3 Software Quality Trade-offs
3.7.1 Interviews ................................................................. 50
3.7.2 Software Metrics ...................................................... 51
3.7.3 Participant Observations ........................................... 52
3.8 Conclusion ................................................................. 52

Chapter 4 - Case Studies ................................................................. 54

Case one - Embedded Software Quality Process ....................... 54
4.1.1 Introduction .............................................................. 54
4.1.2 Overview & Results .................................................. 55
4.1.2.1 Code Review ......................................................... 56
4.1.2.2 Static Code Analysis .............................................. 58
4.1.2.3 Code Churn ........................................................ 59
4.1.2.4 Code Comment Ratio .......................................... 60
4.1.2.5 Software Defects ............................................... 61
4.1.2.6 Request for Requirements to Change ..................... 64
4.1.2.7 Customer Feedback ............................................ 65
4.1.3 Summary .............................................................. 66
4.1.3.1 Inputs into the ESQM-V2 ................................. 67

Case Two - Software Key Practices: Supporting merger and Acquisition .................................................. 68
4.2.1 Introduction ............................................................ 68
4.2.1.1 Overview .......................................................... 69
4.2.2 Software Key Practice Assessments within Software-MNC ...... 69
4.2.3 Internal Assessments as an Integration Framework ................ 70
4.2.4 Prioritising Software Development Practices .................... 71
4.2.5 Learning from Small Team ....................................... 72
List of Figures

Figure 2.1 Example of Factor Identification Table ........................................ 8
Figure 2.1 Embedded Systems Block Diagram (Divekar n.d.) ..................... 9
Figure 2.3 Number of classes and running result (Washizaki et al. 2006) ... 13
Figure 2.4 Embedded Systems Changing Requirements in Phase .......... 15
Figure 2.5 Theoretical Model of Influence on Hardware/Software Interworking (Griffin 2010) ................................................................. 18
Figure 2.6 Team Distributions in GSD .......................................................... 24
Figure 2.7 The Five Levels of Software Process Maturity (Paulk et al. 1993) .............................................................................................................. 27
Figure 2.8 A Software Quality Measurement Framework (Yau et al. 1990) .............................................................................................................. 29
Figure 2.9 Quality Concepts (ISO9126), Trickens et al, 2010 ..................... 30
Figure 2.10 Embedded Software Criteria .................................................. 34
Figure 2.11 Outline of the Initial ESQM ...................................................... 35
Figure 2.12 Initial Embedded Software Quality Model ............................. 37
Figure 2.11 Drechsler Hardware Development Process (Drechsler & Breiter 2004) ................................................................. 42
Figure 3.1 Research Activities and Phases .................................................. 46
Figure 4.1 Case Study One – ESQM-V1 Representation ......................... 55
Figure 4.2 Comms-River Code Review Category ...................................... 58
Figure 4.3 Comms-Lake Code Review Category ........................................ 58
Figure 4.4 KLOC Reviewed for Comms-Lake & Comms-River ............... 58
Figure 4.5 Comms-Lake Defects Discovered Through Review ............... 58
Figure 4.6 Static Analyses for Comms-River & Comms-Lake ..........................59
Figure 4.7 Code Churn for Comms-River ..................................................60
Figure 4.8 Code Churn for Comms-Lake ....................................................60
Figure 4.9 Code Comment Ratio Comms-River ..........................................62
Figure 4.10 Code Comment Ratio Comms-Lake ...........................................61
Figure 4.11 Closed & Rejected Defects -Comms-River ..................................63
Figure 4.12 Closed & Rejected Defects Comms-Lake .....................................63
Figure 4.13 Released Defects Comms-River ................................................63
Figure 4.14 Released Defects Comms-Lake ................................................63
Figure 4.15 Customer Respondents Comms-River .......................................67
Figure 4.16 Case Study Two Embedded Software Quality Framework ......68
Figure 4.17 Case Study Three Software Quality Model Representation ....74
Figure 5.1 Key Area Working Relationships .............................................83
Figure 5.2 Complex Design Key Area Interaction ........................................85
Figure 5.3 Recent Growths in Embedded Systems (Date & It 2014) ............86
Figure 5.4 Customer Focus Key Area Interactions ......................................86
Figure 5.5 Risk Key Area Interactions .........................................................87
Figure 5.6 Industry Standards Key Area Interaction .....................................88
Figure 5.7 Abstraction & User Interface Key Area Interaction ..................89
Figure 5.8 Prototyping Key Area Interactions .............................................90
Figure 5.9 Requirements Definition Key Area Interaction ..........................91
Figure 5.10 Responding to Changing Requirements Key Area Interaction 92
Figure 5.11 Maintainability & Reusability Key Area Interaction ...............93
Figure 5.12 Testing Key Area Interactions ..................................................94
List of Tables

Table 1 Hardware Factors - ESQM ..............................................................38
Table 2 Marketing Factors – ESQM ............................................................39
Table 3 Business & Organisational – ESQM .............................................40
Table 4 Small -Team Key Practice Analysis Results ..................................70
Table 5 FPGA Telco Team ......................................................................77
Table 6 Embedded Software Quality and Value Model ............................97
Chapter 1 - Introduction

1.1 Research Objective

The objective of the research is to investigate the factors that influence embedded software quality in a complex globally distributed development environment. The focus of this research is based on Software-MNC, a multinational company with an office in Ireland.

1.2 Background

The motivation for this research is to understand the factors that contribute to embedded software quality. Just what is embedded software quality? A measure against predefined metrics? Customer collaborations tailoring embedded software guaranteeing the best performance for a customers product? Embedded software that has been commercialised and easily adopted into an organisations next generation product?

Embedded software is expected to do all of the above and more. Embedded Software has been typically used in the hardware industry as a ‘loss leader’ is now a key component in selling and delivering solutions to customers.

There is a difference between embedded software and standalone application software. Embedded software must work with predefined hardware and has strict power and performance requirements. There are similarities in the challenges faced by both embedded software and standalone software. Unlike standalone software embedded software is developed to work with other components within a system.

‘Transformation’ is a word that cannot be escaped when it comes to talking about the technology industry. The Internet of Things (IOT) revolution has brought major change in all areas of the industry, healthcare, automotive, and
aviation. There is strict industry regulation in how devices should be produced especially in healthcare and automotive industries. Embedded systems producers need to guarantee that the physical product and the embedded software are of the highest quality. Neglecting embedded software quality can have adverse effects on the overall company’s reputation.

### 1.3 Motivation

My own personal interest in Embedded Software Development stems from having worked in industry for 5 years. During this time I have held roles as a developer, software quality engineer and project manager. Through each of these roles I have observed how embedded software quality is viewed in that role and how other stakeholders see it. In carrying out this research I identified that developing quality embedded software requires an investment from the entire organisation.

### 1.4 Summary of Research

Software-MNC is a semiconductor company that has several offices based in Ireland. Software-MNC has a presence in 63 countries. Software-MNC produces products for the data centre, automotive, healthcare, telecoms and IOT. Software-MNC has been traditionally viewed as a hardware company but has had to change with the market to provide solutions, which include both embedded software and standalone software. Key capabilities in Software-MNC Ireland include Software Design, Hardware Design, Architecture, Project Management and Technical marketing support. The nature of the research in Software-MNC centres around projects based in the telecoms industry, which is going through a major transformation. Operators and telecom equipment manufactures are looking for new ways in which embedded systems and software can solve some of the challenges they are facing.
The research presented in this thesis is concerned with investigating the factors, which specifically contribute to embedded software quality and value. It examines the relationship between software development, digital hardware design and technical marketing. The relationship between each is then looked at in the context of the business and organisation in which it exists. It defines the factors that software, hardware, marketing and the organisation need to collaborate or co-design when developing an embedded systems product.

1.5 Ensuring Quality & Value

Ultimately organisations undertake programs to create wealth. However organisations want to produce products that fulfil market needs. Meets customer requirements and demonstrate that valuable products with a comprehensive level of quality can be introduced into the industry.

There are many ways in which organisations look to achieve their goals when designing a product.

- Team location
- Organisation structure
- Technical competence
- Technology in use

There are multiple facets that need to be understood in a project team and at an organisational level to ensure an embedded systems product is successful. Given the complex nature in designing and productising embedded systems, the factors that ensure success need to be understood in greater detail and are core beliefs in how the organisation operates. The research focuses on creating a model for embedded software development the Embedded Software Quality Model (ESQM), as there was no embedded software quality model available in the literature. ESQM-V1 was developed as an initial conceptual model. The ESQM-V2 was a refined version of ESQM-V1, which provides a first step in closing the gap.
The ESQM-V2 is a conceptual framework, which is a combination of outputs and learnings from a comprehensive literature review and three case studies in the areas of embedded systems and embedded software development. Through using the Research Methods described in Chapter 3, the ten factors presented in the ESQM-V2 were identified, and are based on theory and practice.

1.6 Limitations of the Research

The ESQM-V2 is a conceptual framework, which has not been implemented. The three case studies were all completed in the Software-MNC. The findings that contributed to the model may not be accurate or the most important should the model be implemented in another company. Implementation of the ESQM-V2 through action research would allow for the model to be verified and updated to truly reflect an embedded systems and embedded software development environment.

1.7 Research Bias

Bias is one factor that makes qualitative research much more dependent upon experience and judgement than quantitative research (Exploable 2012). Interviews and observations were used in two of the three case studies. Of the fourteen participants interviewed only one was female. Candidates were selected for interview based on their role and experience. The technology industry typically has a 20% to 25% female workforce. It is not always possible to have an appropriate gender balance when selecting interviewees. Interviewees selected in the first case study were white Caucasian men all located at an office of Software-MNC located in Ireland. Although selected for their roles an effort could have been made to include one of the senior developers located in China. The use of observations, while was a rich tool to gather information on team dynamics largely excluded the Chinese teams. During the programme there was a whole team onsite
meeting. This was held in the Ireland Software-MNC office, which may not have given a true reflection of team dynamics. As an industry based researcher I had developed assumptions with regards to best practices in embedded systems development. In order to dispel some of these assumptions, the first step in my research was to complete a literature review in order to understand what common problems existed in the industry.

1.8 Structure of the Thesis
This research is presented in four parts,

- **Chapter 2 Literature Review** – This chapter is divided into four sections, covering embedded systems, development stress, global software development and software quality. The Embedded Software Quality Model (ESQM-V1) is introduced, including the five key areas required for successful embedded software development.

- **Chapter 3 Research Design and Methodology** – This chapter outlines the research methodologies used during the construction of the ESQM-V2 and the methods used during the case studies.

- **Chapter 4 Case Studies** – This chapter discusses three case studies undertaken within Software-MNC.

- **Chapter 5 Findings** – This chapter presents the ESQM-V2 derived from learnings from the literature review and the case studies.
Chapter 2 – Literature Review

2.1 Introduction

An initial literature review was conducted examining general software quality factors. A gap was identified in the embedded software quality domain specifically when developing embedded systems. Throughout this chapter I will introduce the factors that are important in the development of embedded software and some of the early Embedded Software Quality conceptual models, which were derived, in the early stages of the literature review. This chapter will conclude with the presentation of the initial Embedded Software Quality Model.

The development of embedded systems and embedded software hinges on many factors. The IOT industry is under going enormous growth. There are new ‘smart products’ being launched in the market. These products are mostly embedded systems. Development stress as presented by Griffin 2010 outlines how multiple factors can have a negative impact on the embedded system development process. Global software development is just one factor that has a significant impact on project success. In order for a project to be successful development stress and global software development need to be understood.

2.2 Literature Review Structure

The hardware and software development processes were examined and their relationship with one another. Given the tight coupling in embedded systems development it was important to understand the interactions that occurred between both engineering functions. During this investigation other key areas and activities came into focus that influenced the interaction between hardware and software, global software development and development stress were two areas that were investigated in more detail.
Miles & Huberman (1994) present that a conceptual framework:

“…explains, either graphically or in a narrative form, the main things to be studied-the key factors, construct or variables, and the presumed relationships among them”

The Four areas identified were repeatedly mentioned as having an impact on the embedded software development process:

- Industry Changes
- Business and Organisational
- Marketing
- Hardware

Each area inputs requirements to the software engineering team. Stakeholders from the four areas expect a certain level of quality and functionality to be delivered through embedded software.

A search was completed for each of the key areas listed above in relation to embedded software quality. The researcher wanted to gather how each of the key areas viewed embedded software quality. And what were the most important factors that needed to be considered in embedded systems development.

An in depth literature review was conducted, examining embedded software quality factors as this was identified as the fifth key area in the embedded system development process. Literature was selected if there was a reference to how the particular area comprehended software quality for embedded systems. There were a limited number of articles discussing the changes in the IOT industry. Online resources such as The Harvard Business Review, The Economist and EE Times were more valuable than current resources when researching the Industry Changes key area.

Due to the gap in the literature for embedded software quality the search was extended to include all software quality factors not specific to embedded
software quality. This included a wider set of papers and ensured that the most important factors were included.

Literature was divided into five categories representing the five key areas. The literature selected in each category was then assessed for the individual factors that contribute the embedded system and software development process. Figure 2.1 shows an example of how the factor analysis was completed.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Financial Strategies</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Customer Strategies</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Learning &amp; Growth</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.1 Example of Factor Identification Table

Throughout this chapter I will demonstrate how the ESQM-V1 was developed through Discussing embedded systems, development stress, global software development including mergers and acquisitions and software quality.

Section 1 Embedded Systems

2.3.1 Embedded Systems

An embedded system is built to complete a precise task that a general computer cannot complete which combines both hardware and software. Embedded systems are becoming larger, more complex increasing in size, have greater quality demands and need to include higher reliability (Rong et al. 2011; Liggesmeyer & Trapp 2009; Washizaki et al. 2006). Constraints on
both the predefined hardware and software exist along with cost, performance, power consumption and weight. Products have been targeted in markets where strict standardisation is in place such as medical devices, automotive and telecommunications (Rong et al. 2011; Sangiovanni-Vincentelli & Martin 2001). Developing a successful embedded systems product that incorporates quality is considerably complex. As more functionality can be moved into software (Sangiovanni-Vincentelli & Martin 2001) it is becoming more difficult to verify the entire design.

![Figure 2.2 Embedded Systems Block Diagram](image)

### 2.3.1.1 Growth in Embedded Systems

In 2000 Sgroi et al stated that by the year 2002 consumers would be purchasing more information appliances than PC’s. The Internet of Things has become a global technical phenomenon. It is estimated that by 2020 the market will be worth 300 billion dollars (Press 2014). Smart cars are expected to create revenue of 400 million dollars and two thirds of homeowners will have purchased connected technology. While traditional embedded systems will continue to exist there is a rapid growth in new platforms. Technology is being used in a way that was never envisioned. Platforms like Intel® Galileo and Raspberry Pi have allowed for enthusiasts to create embedded systems.

### 2.3.1.2 Embedded Systems Design

All system design not just embedded system design will incur trade-offs during the development lifecycle. The complexity of the components
required for embedded systems will have a large range of design choices (Pimentel et al. 2006). Complexities are often introduced in the early stages of the project during the analysis and design work. Mapping architecture is an essential step in the development process as concepts are implemented (Sgroi et al. 2000). Platforms are characterised by programmable components (Sangiovanni-Vincentelli & Martin 2001). How these programmable components fit and work together needs careful consideration. Multiple API’s, programming languages, proprietary operating systems and complex debugging tools equals a heterogeneous system architecture. There are two main ways in which embedded platforms can be designed: a holistic high-level view: or a separation of components that would allow a greater ease of assembly (Sgroi et al. 2000; Sangiovanni-Vincentelli & Martin 2001).

2.3.1.3 Embedded Systems Process

The diverse range of embedded systems products further adds to the complexities in creating a unified development process to ensure success. Each market has different quality requirements and complexities. In the embedded sector, performance, reliability and timeliness are important (Eklund & Bosch 2014; Ard et al. 2014; Kienle et al. 2012). With automotive and medical software lives could be put at risk if a high level of embedded software quality is not achieved, validated, verified and defects resolved as key factors in any embedded system development process (Eklund & Bosch 2014; Trienekens et al. 2010). Weight factors and tailored quality models need to be developed per project. One-size fits all approach will not ensure the correct level of quality across varying types of software projects. Embedded software quality needs to be planned upfront.

2.3.2 Role of Software Engineering in Embedded Systems Development

Software engineering’s role in embedded systems is becoming more important (Liggesmeyer & Trapp 2009). Software can be a key differentiator;
more functionality can now be implemented in software. Hardware redesigns are costly, software is easily changed and can be a less costly option should there be a critical defect in the product, scope creep or redefinition of requirements. Embedded software developers can often neglect goals and constraints like memory footprint and power consumption (Sangiovanni-Vincentelli & Martin 2001; Rong et al. 2011); which they would normally not do when developing stand-alone software. Both hardware and software developers can often claim that the other was responsible. Sangiovanni-Vincetilli 2001 argues that software has paid little attention to hardware requirements and the tight constraints to which software engineering need to adhere. Embedded software requires high quality in order to meet customer requirements and to achieve product competitiveness (Jung 2009).

2.3.3 Embedded Software Complexities

A software team inherits constraints before a single line of code is written. The 2014 Embedded Market Survey (EE Live 2014) highlighted that it is becoming more difficult to meet project deadlines. In previous years this was trending downwards.

The debugging process is close behind project schedules, C is still the chosen programming language for embedded software. As C is an older language tools to debug C are not as advanced as tools for newer languages. Newer languages are developed using more comprehensive development environments. The survey finds that the debugger is the most important tool and an improvement in debugging tools would improve overall design activities. Testing, verification and validation of embedded software can require a more extensive effort (Jung 2009; Di Guglielmo et al. 2012). Investing the effort in a test environment is a necessity given some of the markets into which products are sold. While improving testing and test methods will help ensure the most robust software, it does not ensure project success.
Hardware will often determine the software schedule (Eklund & Bosch 2014). It is not uncommon in projects for requirements to be defined upfront and verified at the end. Embedded software benefits from an iterative design approach, as solutions and goals need to be verified constantly (Jung 2009; Rong et al. 2011). Requirements can be forgotten about during development and it can be late in the process when it comes to light that a requirement has not been met. Changing requirements in hardware can mean that embedded software can become obsolete.

A case study completed in Japan in 2005 examined the correlation between design model quality and the final system performance. Contestants were given a robot as the hardware element and were asked to design software so that the robot would complete an outlined course. Evaluation criteria were based on the design of the software and the overall performance of the robot in completing the course and in time trials. Students were graded from ‘A’ to ‘D’ for their design. Their grade was then plotted against their time trial result, as per Figure 2.3. There was little correlation that the best design will ensure a positive outcome for the overall project (Washizaki et al. 2006). The case study shows how other constraints need to be considered when assessing success for embedded software quality. Seven participants were awarded ‘A’ grades but only two of the participants who were awarded an ‘A’ grade completed the course. These participants finished ninth and fourteenth out of fifteen.
2.3.4 Challenges of the Embedded Software Development Process to Software & Hardware Developers

“Teaching hardware-related concepts to students with a background in software engineering is challenging and vice versa” (Kairus et al. 2003). Hardware and software must be structurally and topologically matched with each other. They can easily come out of sync (Steiner & Athanas 2005). Conflicts will exist given that different stakeholders will define goals (Rong et al. 2011). Conflicting decisions on where functionality should be implemented, performance issues between the hardware and the software algorithm are often discussed. Hardware and software teams must co-operate during development. Co-design can play a role to align both teams requirements and expectations (Rong et al. 2011; Steiner & Athanas 2005). Multiple issues can arise when teams who have worked independently during development enter the integration stage (Ard et al. 2014; Sgroi et al. 2000).

2.3.5 Complexities in Embedded Systems Development

An embedded system is a complex project for an organisation to undertake. There are both technical and organisational challenges in bringing embedded systems to market. Technical complexity is present in multiple areas in an embedded systems program, hardware complexity, software
complexity, understanding of real-time operating systems and the eventual integration of each component. The technical challenges are constrained by the organisational targets and market requirements. Products are developed in a complex global environment that is multi-cultural. There is not only a geographical cultural differences. As mergers and acquisitions continue to increase, further dimensions of organisational complexity are introduced. These factors affect the overall software quality, the capability to meet quality requirements and the customer’s perception of embedded software quality.

Architects provide a key role in overseeing the design of the product. Most project teams are comprised of hardware and software architects as well as engineers. The main role of a software architect is to understand the software implementation. However a SW architect must also understand the hardware design and constraints so that the software can be designed accordingly. Software architects may have a better understanding of the goals and objectives for the product, as they are required to have an understanding of the whole system.

Figure 2.4 is an early version of the ESQM that demonstrates how the software and hardware engineering organisations develop an embedded system. It illustrates how hardware and embedded software milestones are in sync for milestone 1. A change in requirements in “Development Phase 3” which were not relayed to the embedded software development team results in milestone 2 being out of sync. The requirements engineering process can influence the software quality. Changes to requirements in the hardware development lifecycle need to be included during embedded software development. Even though it is accepted that it is easier and cheaper to change embedded software over hardware, changing requirements in hardware without consideration for how it affects software can have a negative effect on quality. Embedded software development needs to be in
sync with hardware development and requirements changes impact the entire development team.

![Figure 2.4 Embedded Systems Changing Requirements in Phase](image-url)

**Section 2 - Development Stress**

### 2.4.1 Project Success

In general project success cannot be defined by a single common goal (Toor & Ogunlana 2010; Ebbesen & Hope 2013; Jha & Iyer 2007). Stakeholders will have their own definition of project success. Within each stakeholder’s definition of success there are two levels of success - macro, long-term strategic success, and micro, short term operational success (Toor & Ogunlana 2010). Project success can differ depending on the involvement of the stakeholder and the stage at which they get involved (Turner et al. 2012; Ebbesen & Hope 2013; Toor & Ogunlana 2010; Jha & Iyer 2007). An understanding of the organisation is a core starting point in delivering the correct services to the organisation (Solichah et al. 2013).
2.4.1.1 Project Success Criteria

A single set of factors cannot be identified to guarantee project success or ensure that all stakeholders interests are being met. ‘One important aspect of the project management is to acquire the information related to an optimum balance between the projects objectives’ (Razavi Hajiagha et al. 2013). Numerous studies have examined what the “Modern Project Constraints” are across multiple disciplines: some constraints are environmental, organisational or linked to skills and experience.

Softer critical success factors that are not easy quantifiable influence project success. They are strongly embedded in organisation culture (Basu 2014). The contributing activities, the overall business objectives and the stakeholder’s interests must be accounted for and understood to have a complete picture of project success.

Softer critical factors include:

- Top Management Commitment and Support
- Owners Competence
- Project Managers Competence
- Monitoring Feedback
- False project conceptualisation
- Doing the right thing

2.4.2 What is Development Stress?

Development stress as outlined by Griffin 2010 is:

- Any cause of developer dissatisfaction or negative experience with the product or service designed;
- Any aspect of inter-team relationships that causes an adverse material change to the project plans and market aspirations;
- Any negative influence on developer productivity.

The existence of development stress in a product can potentially have a negative impact on the product and could deem the product a failure. Weiss
outlines how changes in productivity, quality, effort and other characteristics are affected by a multitude of factors including the development process, technology, management focus, and market place shifts and staff turnover (Weiss et al. 2002).

2.4.2.1 Managing Development Stress

It is unlikely that any project will escape without some development stress during the product development lifecycle. Project Management is the application of tools and techniques (Atkinson 1999) that empower the project manager in making a decision during the project that will impact positively or negatively on project success. “Good project management requires a relevant set of performance metrics” (Marques et al. 2011). As discussed no one set of metrics can be applied and the iron triangle is not a sufficient decision support model. Goals can often be identified at the start of the project and ignored during development phases (Rong et al. 2011). Time must be invested at project initiation in determining and tailoring the Key Performance Indicator (hereafter KPI) and Critical Success Factors (hereafter CSF) that are fundamental to the decision support model for that project given that a project is a “unique process”. Project managers need to be adaptive to their environment (Ebbesen & Hope 2013).
2.4.2.2 Development Stress Model

Figure 2.5 Theoretical Model of Influence on Hardware/Software Interworking (Griffin 2010)

Figure 2.5 Theoretical model of influence on Hardware/Software networking, (Griffin 2010) outlines how business objectives and market demands influence hardware and software development teams. As outlined in the model, hardware is not easily changeable while software is viewed as being easily changeable. There is more focus ensuring the requirements for hardware are defined upfront, requirements matrix are generated so that verification and validation can be completed. The objective of software in an embedded product is to be a differentiator. Requirements for embedded software often follow the hardware requirements and may not be as well defined or thought out. The flexibility that software provides can compromise software quality. Embedded software products are developed within market constraints and business objectives. A product is designed to fulfil these constraints. A company might delay a product launch if the hardware has not been fully tested and verified, however they may not delay a product launch if the embedded software has not been fully validated and verified. Hardware and embedded software quality are assessed and quantified differently.
2.4.2.3 Mitigating Development Stress

A project must look to incorporate all its stakeholders requests, objectives and goals. As discussed in 2.4.1 stakeholders will have their own definition of project success. These may change during the project, which can further increase development stress, but formal way of accepting changing goals needs to be built in. Stakeholders need to be mindful of each others goals and objectives. Compromise is necessary in order to achieve the best possible result for all stakeholders. Goals and objectives need to be verified constantly and revisited as a whole team to ensure that everyone in the project has a common understanding of what needs to be achieved from the project. (Rong et al. 2011)

2.4.3 Inputs to the ESQM-V1

There is no single metric that can measure development stress in any project. Successful project teams are not lucky - they are dedicated in their pursuit in delivering the best project. Team members must have trust and respect for one another, and in the skills and knowledge that each person brings to the project. Development stress stems from a misunderstanding in how the other team works, the software engineer not understanding the Business case or the marketing team not giving detailed requirements to the hardware team.

The ESQM-V1 therefore needs to identify the areas where development stress can originate in embedded systems development.

Section 3 - Global Software Development
2.5.1 Why Global Software Teams?

Global software development (GSD) has been in existence since the 1970’s, it is unlikely that any software team co-located (Cataldo & Nambiar 2012). The idea of “follow the sun” development has many benefits for the company. “Cost advantage and a large labour pool” (Carmel & Agarwal 2001) are viewed as the primary reasons.

However there are multiple reasons for such teams.

- Physical closeness to the customer
- Understanding of the culture
- Cater to local markets
- Access to skills that are not available
- Cater to local Markets
- Round the clock development

2.5.1.1 Challenges of Global Software Development

There are four key contributing factors when operating in a global distributed team, (Lee et al. 2013; Richardson et al. 2012; Verner et al. 2014).

- Geographical Distance
- Time Zone
- Culture
- Organisation

2.5.1.2 Communication and Coordination

Communication and coordination need to be maintained in all projects. The greater the prevalence of each of the four factors increases the difficulty in managing communication and coordination (Scott 2013) and lead to difficulties in practicing strong software development methodologies. Ensuring process rigor and standardisation is a difficult task. (Lee et al. 2013). A team should be working towards a common goal and have the same objectives (Allan 1997). Global teams should not differ from traditional teams in working towards the same goal and objective. This is not always the case as geographic dispersion can lead to less information sharing and a lack of
respect and trust – all of which can negatively impact on communication (Noll et al. 2010; Scott 2013). Often the formal channels of communication are too slow in global teams this causes information to be propagated informally (Herbsleb & Mockus 2003). This can have a negative effect on the team, as not all team members will receive information in a timely manner, thus causing more issues in trust and cooperation.

2.5.1.3 Team Distribution

The distribution of the team is a key factor to consider when reducing the negative effects GSD teams incur. The distribution of the team members in the respective sites is a key-contributing factor. The more evenly distributed the team the greater the effort in maintaining commitment and coordination, as each team member in their respective sites has to put additional effort into maintaining a communication path (Lee et al. 2013). There are human factors that need to be considered, trust, fear and teamness. (Richardson et al. 2012) A team needs to be productive and functional to complete the project but there is also a social element, allowing the team to get to know one another outside the work environment.

2.5.2 Mergers and Acquisitions in GSD

Mergers and Acquisitions (M&A) are used strategically in a GSD environment. There is an advantage in choosing to acquire or merge with a company. There is an existing customer base, reporting and team structures are in place.

A company that is being merged or acquired may have to remain in its current location thus becoming an additional location. Multinational companies often purchase start-ups, smaller companies or divisions of companies as a way to gain a foothold in a particular market segment or reduce the risk of a potential threat. Ultimately they are motivated by value creation opportunities (Cartwright & Schoenberg 2006; Meyer 2008; Colman
Acquisitions can facilitate accelerated growth in a market segment in which the company is struggling to grow organically, or allow the company to obtain a disruptive technology that could reduce market share. Acquisitions are undertaken to encourage all stakeholders - suppliers, customers, employees, communities, managers and shareholders - benefit over time (Freeman et al. 2004).

While acquisitions bring expertise, technology and potential market share most acquisitions fail and there is often a low retention of staff from the acquired company. The combination of new people and technology creates an environment of continual team re-formation (Weiss et al. 2002). Additional ‘stress’ (Nahavandi & Malekzadeh 1988) is placed on the organisations as management structures are merged, objectives are aligned and processes adapt. Much of the literature focuses on the human and top management support that needs to be in place in order to complete a successful M&A. This is well-researched topic as the number of M&A’s has increased year on year.

M&A’s can create a hostile and uncertain environment in the company that will be acquired. Employees in the acquired company may feel hurt, anxious, angry and disappointed (Seo 2005) or seek to retain autonomy. The M&A process needs to understand the integration process for:

- People
- Technology
- Process

It is important that the acquiring company treat the acquired company with fairness and respect (Meyer 2008). It is in the early stages of integration that allow for opportunities to learn from the acquired company on their successes and ‘challenge old ways of doing things’ (Colman & Lunnan 2011).

2.5.2.1 Merger and Acquisition Integration Stages

Seo and Hill outline a four phases in the M&A integration stages,
1. Premerger
2. Initial planning and formal collaboration
3. Operational combination
4. Stabilisation

Other authors refer to pre-acquisition, integration process and post-acquisition. It is important that organisations are cautious in planning integration activities. Cultural differences will emerge during the entire process. Careful consideration in the “Initial Planning and formal collaboration” stage needs to be given to how these addressed and in which phase of the process (Weber & Tarba 2012). The organisation needs to be looked at as a whole, how subunits work together, creating better synergy between people, process and technology (Barkema & Schijven 2008). M&A’s occur to create wealth but customer commitments and the business objectives cannot be ignored during an M&A. Organisations are vulnerable during the M&A process. The best time to attack your competitor is in the middle of a merger (Meyer 2008). Both organisations must communicate, cooperate and coordinate in ensuring that business at hand is receiving the proper attention while planning how the integration process will complete successfully.

2.5.3 Complexities in GSD Communication
Figure 2.6 Team Distributions in GSD

Figure 2.6 demonstrates how differently distributed channels of communication exist based on the dispersion of the team. In a development team of four people there will be twelve individual channels of communication. The blue arrows signify communication between team members that are collocated, the orange arrows signify communication between team members that are not collocated. In GSD Team 1 66% (eight of the twelve channels) of communication will occur over email, phone or instant messenger. However in GSD Team 2 where there is an unequal distribution of team members only 50% of its channels of communication will be completed via technology. A greater effort is put on the team to ensure that there are open channels of communication. According to Lee et al 2014 GSD will be more successful if more of the team is collocated. Successful GSD is possible but a number of factors have to be considered in how the team is setup and all risks should be captured with risk mitigations in place.

2.5.4 Difficulties in M&A and GSD

Effort needs to be invested in planning the integration process. There is no model that illustrates how to integrate embedded software teams. The lack of such a model results in poor integration of teams and the loss of knowledge and skills (Meyer 2008). This is the opposite to the goal of any M&A as organisations look to create knowledge. The company completing
the M&A must give the company respect. Respect for the technology it has produced, the processes it has used to launch a new product and for the staff who will need to adapt to a new company and its culture. The company completing the M&A needs to invest time communicating why it chose to purchase the company and the value, knowledge and markets they will be ing with them. Seo and hill outline the need for a premerger stage to address some of the issues that might arise when a merger is in progress. Organisations want to make use of new teams skills and knowledge. Like development stress teams need to be conscious of other influences that may affect how new employees may approach a project. Respect, communication/coordination and trust are important in creating an environment that will enable a successful merger. Mergers and acquisitions are another form of GSD, teams developing embedded systems should show equal respect and trust to one another. As figure 2.6 demonstrates communication and coordination requires significant more effort in GSD teams. All team members need to be aware of the additional effort required when working in a GSD environment.

Section 4 - Software Quality

2.6.1 Quality

Quality as defined by the Oxford English Dictionary,

“The standard of something as measured against other things of a similar kind; the degree of excellence of something”

Quality might mean:

- Standard
- Characteristic
- Essential Property
- Excellence
Quality as a description on a product, service or process encompasses both a minimum target of what is expected and higher objectives that have connotations with something of excellence. Users’ perception about product quality varies across user types (Jung et al. 2004). Expectations regarding quality can be different.

2.6.1. Quality Assurance

Assurance as defined by the Oxford English Dictionary,

“A positive declaration intended to give confidence; a promise”

A company that labels their product, process or service as a quality assure, by the definition of assurance, means that they are standing over their defined quality standard for the particular product, service or process.

2.6.1.2 Quality Standards

Standards facilitate minimum acceptable quality levels in most industries. The Capability Maturity Model and ISO are widely recognized and implemented standards used in software development. Both of the standards outline a generic way in which quality or maturity can be achieved during software development. Paulk describes the CMM as being intentionally abstract capturing “universal truths” about high performance software organisations (Paulk 2001).

The International Standards Organisation (ISO) was setup in 1974 to outline world-class specifications for products, service and systems, to ensure quality, safety and efficiency (ISO 2008). ISO 9001 is an international quality management standard, which can be applied to many businesses. The ISO have revised this document and updated the standard, this is a reflection that the term quality and quality assurance can change.

There are eight quality management principals that make up the ISO 9000 (ISO 2008).
- Customer Focus
- Leadership
- Involvement of People
- Process Approach
- System Approach to management
- Continual Improvement
- Factual approach to decision making
- Mutually beneficial supplier relations

ISO 9000 provides a generic guidance on the key benefits of each principle and how to apply it within the organisation. In order to assume success with the principles knowledge of customer goals, organisational direction, processes and market direction are required (ISO 2012).

The Capability Maturity Model is a five level model that describes best practice engineering and management techniques (Paulk 2001). Organisations are assessed and awarded a CMM level from 1 to 5 within the model, level 1 being the lowest and level 5 being the highest. There are 52 goals and 52 key process areas or KPA’s.

![Figure 2.7 The Five Levels of Software Process Maturity](Paulk et al. 1993)
**Level 1** initial focus on competent people. This sets the baseline in the organisation that it can satisfy the basic CMM guidelines.

**Level 2** Repeatable focuses on project management process. Project successes and failures can be understood within the organisation. This allows for the organisation to guarantee a level of project success through its documented processes.

**Level 3** Defined focuses on engineering process and organisational support. Organisations want to hire and retain the best people so that a quality product can be developed. Investing in the organisations engineering skills is one way to guarantee a quality product.

**Level 4** Managed focuses on Product and process quality. Level 2 and 3 focus on process and engineering separately. Level 4 combines process and engineering, it is only by understanding the whole process can the best product be produced.

**Level 5** Optimizing focuses on continual process improvement. There is always room for progress in the organisation. Process and engineering optimizations allow for a company to stay competitive.

The levels in the CMM are defined that it takes a project view in level 2, organisational view in level 3, a quality view in level 4 and a reaction or ability to handle change in level 5.

### 2.6.2 Software Quality

IEEE Standard 1061-1988 defines software quality as the degree to which software possesses a desired combination of attributes (IEEE 1998). Once the software possesses a desired combination of attributes it is deemed to have some level of quality. The desired attributes can differ from company to company. Software quality is generally defined and measured by metrics.
These allow classification, comparison and quantitative analysis to be applied (Yau et al. 1990).

### 2.6.2.1 Software Quality Assurance

Software quality assurance is generally guaranteed by the organisation supplying the software. Software quality teams will ensure that standards have been adhered too. Many companies will use CMMI, ISO and other standards as a key foundation for the software quality metrics that they define and measure against. Figure 2.8 shows a typical software quality model in which each of the factors are measured against quality criteria. Software quality assurance ensures that the process and requirements outlined are followed during the development process and the product satisfies all the criteria (Alsultanny & Wohaishi 2009). Quality assurance increases confidence in the product that is being delivered to the customer.

![Figure 2.8 A Software Quality Measurement Framework](Yau et al. 1990)

### 2.6.2.2 Software Quality Requirements for Embedded Systems

In any project there are functional and non-functional requirements. This is true for software quality too. Boegh outlines that there are two different viewpoints:
1. Satisfaction of requirements, according to specifications
2. Satisfaction of stated and implied needs, fit for purpose (Boegh 2008).

![Quality Concepts (ISO9126), Trickens et al, 2010](image)

The viewpoints presented by Boegh, highlight that products are built in the right way or does. When reviewing literature to define software quality requirements authors take either viewpoint. ISO 9126 outlines a difference between ‘quality in use’, ‘external’ and ‘internal quality’ (Trienekens et al. 2010). The lists below outline some of the characteristics of internal and external software quality.

Built according to the specifications and internal quality:

- Defect count
- Complexity
- Tools used
- SLOC
- Meet Required Standards
- Unit Testing
- Integration Testing

Fit for Purpose and external quality:

- Performance Testing
- Changeability
- Maintainability
- Customer Satisfaction
- Meets Market Criteria
- Fulfils organisational Objectives

Software quality teams outline metrics that are to be achieved based on expert opinion (Kläs et al. 2010). Software quality is a key factor in the software
development process (Trienekens et al. 2010; Kläs et al. 2010). While software quality metrics have been successfully adopted and implemented in embedded software (Oliveira et al. 2008). While some standards outline embedded software quality practices like ISO 26262 (automotive) and DO-178C (avionics) there was no single model identified in the literature for embedded software quality. Embedded software quality teams should take an approach that merges specifications and fit for purpose. Early definitions of software quality aimed to define a single metric for all software projects for complexity, defects and various other empirical studies. However this did not provide the single bullet to define software quality requirements (Kläs et al. 2010). Griffin 2010 presents a model that outlines development stress comes from many areas, which will affect the software quality. Embedded software quality teams must take a systems approach to incorporate all areas of development.

2.6.2.3 Software Quality Trade-offs

Like all decisions that are made during product lifecycle, trade-offs exist in software quality. Software quality is subject to meeting business needs: software quality must be defined accordingly (Alsultanny & Wohaishi 2009). The desired outcomes of quality are that it increases revenue, reduces costs and improves productivity. Ensuring quality can absorb use 30% to 90% of the project budget (Kläs et al. 2010).

2.6.2.4 Where are Software Quality Issues Introduced?

It is well accepted that most problems in software development are introduced in the early phases of the project (Boehm 2000). Requirements engineering, analysis, design and architecture phases of the project play a vital role in defining product success (Rahman 2013). The role of architecture is to take a system wide approach when designing an embedded systems product,(Cui 2010). I refer to the model developed by Griffin(2010) which outlines where ‘stresses’ are introduced when designing embedded systems and the functional areas of the organisation. Thus architects or architecture
teams should incorporate all stakeholders requirements when designing a product. To meet customer expectations, the design process must begin early in order to understand the extent that customer expectations can be met (Croll et al. 2008).

IEEE Std 1471-2000 (IEEE 2000) defines architecture for software intensive systems as:

\[ \text{The fundamental organisation of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.} \]

Croll presents an argument that architecture is the foundation of project success and overall software quality that will be achieved by the end product. Software developers will implement features based on the design specifications that have been agreed by the architects (Liggesmeyer & Trapp 2009). These specifications will or should include the exact requirements that are to be delivered to the customer such as performance, complexity, maintainability, reusability, tests, etc. Software complexity often increases as the project progresses through the project phases (Pimentel et al. 2006; Alsultanny & Wohaishi 2009). In the early phases architecture should explore a wide range of design choice to assess trade-offs and determine the best solution for delivering the system to the customer and which incorporates the most valuable software quality practices. Interviews conducted by Weiss demonstrate the differing opinions on software quality from engineering who believe that the product should wait to achieve optimal quality and marketing believe that it should be released to the market so as not to miss an opportunity (Weiss et al. 2002).

2.7.1 Embedded Software Quality Complexities

Standards are necessary in providing guidelines when developing embedded software. ISO 9126 (Jung et al. 2004) is not specifically designed
for embedded software quality. ISO 9000 outlines quality management factors, which can be used in any industry. Implementing both these standards during embedded software development can have a positive effect on the overall embedded software quality. However, there is no guarantee that the desired level of embedded software quality will be produced using standards alone. Although organisations that develop embedded software quality invest time in defining internal embedded software quality metrics, these are mostly defined by the software engineering organisation. Embedded software quality requires inputs from the whole organisation in order to ensure the embedded software developed is also valuable. Software quality metrics allow for a completion of a ‘health check’ but they do not reveal the true worth of the embedded software. As mentioned in 2.6.2.2 there are standards specifically for embedded software. The researcher limited the scope of this research to ISO9000 and CMMI.

2.7.1.1 Inputs into the ESQM

![Figure 2.10 Embedded Software Criteria](image)

**Figure 2.10** Embedded Software Criteria
Figure 2.10 shows that there are three main inputs into embedded software quality criteria. Both ISO and CMM/CMMI are standards which in many organisations from the basis for building a software quality process. It is through the implementation of these processes and standards that internal software quality criteria are defined. Software metrics allow for analysis to be completed on the health of the embedded software and can be easily communicated to stakeholders in the program. Senior managers can interpret the metrics and recommend changes to the embedded software quality before it is released to a customer.

**Literature Review Conclusion**

**2.8 Embedded Software Quality Model**

Organisations invest effort in resourcing software quality teams to develop implement and govern software quality. The metrics that have been defined lean heavily on the use of standards, ISO and CMM/CMMI. The existence of such standards has contributed heavily to the development of software quality criteria and frameworks.

There is a requirement for embedded software quality frameworks and criteria. Embedded software is more complex than stand alone software, due to the tight coupling with hardware. The embedded systems market is competitive. The industry, marketing and the organisation recognise that software is a key differentiator.
To complete the literature review, I assimilated the factors identified, and developed the Embedded Software Quality Model Version 1 (ESQM-V1). The ESQM-V1 outlines four key areas that can have an impact on what is traditionally defined as software quality.

- Hardware
- Marketing
- Business & Organisational
- Industry Changes

As software is now viewed as a differentiator by industry analysts and within organisations it is important that factors from the four key areas are incorporated so that embedded software adds value as well incorporating the traditional quality factors.

2.9 Research Questions

Software quality research is generally focused on metrics. There is limited literature focusing on the development of embedded systems and embedded...
software. The main issues this research will highlight are discussed in the form of questions:

1. What are the key factors that define success in the context of embedded software quality?

2. Who are the stakeholders involved in embedded system design?

3. What is the role of teams who are fundamental in embedded systems design in ensuring embedded software quality and value?

2.10 An Initial Conceptual Framework – ESQM-V1

The quantitative data generated from the literature review formed the initial conceptual model (Figure 2.12). Tables 1-3 show how the factors were derived. This allowed for themes to be identified across the key areas in embedded systems and embedded software development. The ESQM-V1 was used as a framework through which a more developed ESQM-V2 was created.
Figure 2.12 Initial Embedded Software Quality Model
<table>
<thead>
<tr>
<th>Table 1 Hardware Factors – ESQM-V1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintainability/Reusability</td>
</tr>
<tr>
<td>Component Assembly</td>
</tr>
<tr>
<td>Complex Electronic Design</td>
</tr>
<tr>
<td>Post &amp; Pre Hardware Validation</td>
</tr>
<tr>
<td>Flexible Implementation Methods</td>
</tr>
<tr>
<td>Architecture Design &amp; Methodology</td>
</tr>
<tr>
<td>Strict Requirements</td>
</tr>
<tr>
<td>System Level Testing</td>
</tr>
<tr>
<td>Development Cost</td>
</tr>
<tr>
<td>User Interfaces</td>
</tr>
<tr>
<td>High Level Abstraction</td>
</tr>
<tr>
<td>Industry Standards</td>
</tr>
<tr>
<td>Component Level Testing</td>
</tr>
<tr>
<td>Separate Communication &amp; Compute</td>
</tr>
<tr>
<td>Changing Requirements</td>
</tr>
<tr>
<td>Technical Documentation</td>
</tr>
<tr>
<td>Structured Development Process</td>
</tr>
<tr>
<td>Table 2 Marketing Factors – ESQM-V1</td>
</tr>
<tr>
<td>-------------------------------------</td>
</tr>
</tbody>
</table>

| Customer Requirements | \( \times \) | \( \times \) | \( \times \) | \( \times \) | \( \times \) | \( \times \) | \( \times \) | (Croll et al. 2008) |
| Trade-offs | \( \times \) | \( \times \) | (Cui 2010) |
| Customer Expectations | \( \times \) | \( \times \) | \( \times \) | \( \times \) | \( \times \) | \( \times \) | (Weiss et al. al. 2002) |
| Customer Experience | \( \times \) | \( \times \) | \( \times \) | \( \times \) | (Hall & Johnson 2009) |
| Customer Satisfaction | \( \times \) | \( \times \) | \( \times \) | \( \times \) | \( \times \) | \( \times \) | (Gregorio 2012) |
| Early Design Process | \( \times \) | \( \times \) | \( \times \) | (Freeman et al. al. 2004) |
| Fast Paced Markets | \( \times \) | \( \times \) | (Turner et al. al. 2012) |
| Business Strategy | \( \times \) | \( \times \) | \( \times \) | (Patricio et al. al. 2004) |
| Market Development | \( \times \) | \( \times \) | \( \times \) | \( \times \) | \( \times \) | (Karlsou et al. al. 2007) |
| Customer Collaboration | \( \times \) | \( \times \) | (Cléments & Northrop 2001) |
| Industry Standards | \( \times \) | \( \times \) | (Clements & Northrop 2001) |
| Architecture Design | \( \times \) | \( \times \) | (Clements & Northrop 2001) |
| Engineering Direction | \( \times \) | \( \times \) | (Clements & Northrop 2001) |
| Risk | \( \times \) | \( \times \) | (Clements & Northrop 2001) |
| Customer Benefit | \( \times \) | \( \times \) | (Clements & Northrop 2001) |
| Relationships | \( \times \) | \( \times \) | (Clements & Northrop 2001) |
| Political Support | \( \times \) | \( \times \) | (Clements & Northrop 2001) |
|-------------------------------|------------------------|-------------------------|-----------------------|---------------------|
| Strategic Direction           | X                      | X                       | X                     | X                   |
| Financial Strategies          | X                      | X                       | X                     | X                   |
| Customer Strategies           | X                      | X                       | X                     | X                   |
| Internal Business Process     | X                      |                         |                       |                     |
| Learning & Growth             | X                      |                         |                       |                     |
| Understand How Change         |                         |                         |                       | X                   |
| Impacts the Organisation      |                         |                         |                       |                     |
| Trade-offs                    |                         |                         |                       | X                   |
| Risk                          |                         |                         |                       | X                   |
| Forecasting                   |                         |                         |                       | X                   |
| Diversity                     |                         |                         |                       | X                   |
| Shareholder Value             |                         |                         | X                     | X                   |
| Market Share                  |                         |                         |                       | X                   |
| ROI                           |                         |                         |                       | X                   |
| Future Trends                 |                         |                         |                       | X                   |
| Social Responsibility         |                         |                         |                       | X                   |
2.11 Embedded Software Quality Model Version 1

2.11.1 Hardware

Hardware designs are complex. Components often have to satisfy current product needs and fulfil future product requirements. Hardware designers must have a clear definition of system requirements but also respond to changing requirements during the project lifecycle. Hardware development is costly both financially and from a schedule perspective. Many embedded systems programs follow adopt modify process. The adopt modify process allows organisations to reuse components from products that have been developed when developing a new product. Key factors contributing to successful adopt modify implementations include:

- Maintainability & Reusability
- Component Assembly
- Architecture Design & Methodology
- Flexible Implementations
- Separation of Control and Compute
- Technical Documentation

Hardware designers need to assess if components that have been developed in the past can meet the requirements for the new product brief. There is no guarantee that the engineer who developed the original component will be in the company when it is reused. Processes need to be in place to properly document the technical detail that may not be evident from the code.

There is a strong focus on validation, verification and simulation models to ensure that the product will work as intended. This focus on testing is largely due to the cost of hardware re-spins. If a component is to be reused hardware engineers want assurance that the component or components has been through rigours testing, at a component and system level.

There are similarities in hardware and software development models. Both have development processes that are split into phases of design, planning,
development and testing. Hardware development is more defined and structured following more of a waterfall model.

![Figure 2.13 Drechsler Hardware Development Process](Drechsler & Breiter 2004)

2.11.2 Marketing

The marketing team has to walk on a tight rope so that it can deliver what the customer wants but within what engineering can meet given the business and organisation objectives.

They are the key link to the customer, in setting engineering direction and understanding customer requirements. They are tasked with growing a market segment, building brand value and ensuring repeat business by ensuring customers are;

- Satisfied
- Have a good experience of engaging with the company and/or using the company’s IP
- Benefit from choosing the brand
Relationships need to be nurtured both in and out of the company. Obtaining internal organisational and political support is under the remit of the marketer. Business strategy and an understanding of team dynamics at an executive level are often required. Customer and partner relationships are vital when competing in a competitive industry. Collaboration and customer involvement in the early design process can ensure success.

2.11.3 Business and Organisational

“In business: What matters most may be the extreme but rare possibility, not the most likely one” (Sargut & McGrath 2011). CEO’s, VP’s and GM’s set strategic direction and deliver shareholder value through ROI and profit. Depending on the size of the organisation this can be set at a company level or at group or divisional level. Senior management are responsible for defining the organisation structure. Choosing which markets to invest in, the physical location of the offices, financial reporting structures and pursuing mergers or acquisitions. Senior executives need to be aligned where synergies can occur to meet business objectives and where conflicting divisional strategies will impact the business objectives.

There is a corporate and social responsibility. The business must facilitate learning and growth for their employees ensuring staff retention and employee wellbeing. Many tech companies have a diversity and social responsibility. Microsoft commits to serving the needs of people. Google offer grants and are recognised publically for their social responsibility programs (LinenShapeSpace 2015).

2.11.4 Industry Changes

The technology industry is diverse and is constantly changing and evolving. No company can afford to look exclusively inward ignore the evolutions. Nokia lost market share in mobile phones, choosing not to implement touch screens in their devices and subsequently loosing market
Embedded systems are becoming smaller, more complex and have to meet stricter requirements.

Embedded systems designers are evaluating how they can achieve the most efficient design. The required functionality can be completed in either software or hardware. Designers need to understand the trade-offs of implementations in either hardware or software. When designers carefully consider how to best implement software organisations it can be used as a differentiator in the market.

2.12 Summary

There is no one model or “silver bullet” that can ensure software quality in embedded systems programs. Standards such as ISO and CMMI have been used by organisations to better their process that influences software quality. The scope of software quality has evolved. It is no longer just defect count, complexity and test case reports. It is an assessment of its usability and customer interpretation of quality: does it fit their intended requirements? Debugging while one of the most important tools in a developers tool kit, is only one part of ensuring software quality. Given the global nature of embedded software development, and varying practices and process for tools, it is not always the case that one tool and a single process can be achieved, especially when mergers and acquisitions need to be integrated. The organisations goals need to be understood when implementing a model to ensure software quality. Software is one of the main interfaces that the customer has when using an embedded system. A project can be on time, under budget but if it does not fulfil the customer’s needs or market requests it will not make a profit.

Embedded software can often be viewed as a throw away by hardware engineering and marketing teams. Its easy to change, you can get it right in the next release. Hardware engineering teams may not realise that their
attitudes towards software development affect the overall quality. Software quality models include input from standards and software quality experts but there is little focus on incorporating hardware engineer’s criteria for software.

ESQM-V1 was developed through the literature review examining the factors that contribute to embedded systems development and affect embedded software quality. Metrics are useful and important in measuring software quality. More effort is required from all of the embedded systems team to guarantee embedded software quality. The embedded systems market is fast paced: goals and objectives need to be at the forefront of the teams focus throughout the entire development process to ensure that they will satisfy the customer and be successful in the industry. ESQM-V1 is based solely on theoretical findings. As an industry-based researcher I was most interested in refining my ESQM-V1 model by undertaking practice-based research. Case studies were completed in Software-MNC in an effort to validate the findings from the literature. The research methods are detailed in the next chapter.
Chapter 3 – Research Design & Methodology

3.1 Research Overview
Learnings from the literature review formed the basis for the ESQM-V1. Case studies were selected based on their suitability to apply learnings to the Embedded Software Quality and Value Model. Following the literature review and completion of the case studies a final Embedded Software Quality and Value Model was accomplished.

3.2 Research Phases
The research was completed in four stages (Figure 3.1). Chapter 2 outlines how the literature review and ESQM-V1 were completed. This chapter will outline how the case studies were completed and how the ESQM-V2 was developed. The literature review provided the foundations to develop the ESQM-V1. Three case studies were identified to gain an industry-based perspective. These learnings were incorporated into ESQM-V1 to develop ESQM-V2.

Figure 3.1 Research Activities and Phases

3.3 Case Studies
The objective of this research is to understand what the embedded software quality factors are and how divisions outside of software
engineering viewed software quality. Each case study was selected as it represented one or more key areas represented in the Embedded Software Quality Model. The “Software Quality” component was required to be present in each case study. A case study, which encompassed all five key areas, was not feasible. Learnings from each case study could be applied to the ESQM-V2.

3.3.1 Embedded Software Quality Process

Examined internal software quality in programmes that were completed by Software-MNC. A software profile analysis was completed on two programs. Comms-River, a develop program and Comms-Lake an adopt modify program. This case study provided a baseline understanding of embedded software quality and the metrics used in Software-MNC. This allowed for fair comparisons in how embedded software quality was implemented in the remaining case studies.

3.3.2 Software Key Practices: Supporting merger and Acquisition

Assessed software quality practices in Small-Team, which was acquired by Software-MNC, examining how organisational changes affect the software development process. The baseline understanding of embedded software quality in Software-MNC was used as a comparison. This was assessed in order to determine the most valuable embedded software quality practices. Small-Team’s portfolio of products focuses on delivering embedded communications products.

3.3.2.1 Access to the Companies

The researcher was completing a job assignment in the software quality team when the acquisition was in progress. Two colleagues supported the researcher to complete the software key practice assessments in Software-MNC and Small-Team. The case study was completed within the software teams in both companies when Small-Team was in the early stages of integration into Software-MNC. Thirteen of the key practices were chosen as
the basis for the interview questions (Appendix). Eleven participants were interviewed from each of the three teams. Project managers, technical leads and senior management were selected. Each candidate was asked to describe how each of the key practices was done in the team. The data presented in table 4 on Software-MNC is the assessment score from the 2014 assessment, which was conducted by a colleague A in the software quality team. Three assessments were completed during August 2014: the researcher conducted interviews with the teams based in the UK and the U.S. a second colleague B conducted interviews with the China team.

3.3.3 FPGA For Telco Use Case

The team were tasked with developing a prototype embedded systems development kit. The development kit was to enable customers develop next generation products in the telecommunications industry. This case study incorporated four out of five of the key areas as identified in the Embedded Software Quality Model. The aim of this case study was to understand how other key areas impact or assist in embedded software development. The case study incorporated the “Digital Hardware Design” component and the influences of “Business Realities” and “Risks” as outlined by Griffin 2010 and illustrated in Figure 2.5.

3.4 Mixed Methods Design Qualitative & Quantitative

Using a combination of qualitative and quantitative research methods can be very beneficial. Qualitative methods can be used for exploratory purposes, quantitative enables further elaboration and provides order and importance to the qualitative data (Creswell 2013). Mixed methods research combines both research forms. Qualitative hypothesis can be validated through subsequent quantitative work.

Qualitative research describes an approach to understand meanings, beliefs and experiences. It has been defined as “any kind of research that produces
findings not arrived at by means of statistical procedures or other means of qualification” (Hoepfl 1997). Quantitative research examines specific relationships between measurable variables. Quantitative methods were used to measure and compare software quality across multiple projects. Qualitative research was used to understand the environment in which embedded software quality is developed. Mixed methods ensure that a feedback look can be built in. Quantitative approaches were taken to understand the relationships between each of the factors.

3.5 Triangulation

Triangulation refers to the use of more than one approach to the investigation of a research question in order to enhance confidence in the ensuing findings. (Bryman 2011) Using this method facilitates more consistency in overall research findings. Methodical triangulation was used, as both quantitative and qualitative methods were selected. Quantitative methods were used to derive the ESQM-V1. Qualitative methods like interviews and observations provided a level of reliability from the findings in the literature review. The developed ESQM-V2 incorporated data from two data sources literature and case studies.

3.6 Developing the Completed ESQM-V2

The developed ESQM-V2 was derived from the initial ESQM-V1 and the learnings existed in two or more of the five key areas. Learning’s from the case study were crosschecked against the factors in the initial ESQM-V1: this was to ensure that all relevant factors were included. If a factor only existed in one of the key areas in the initial ESQM-V1 but was identified in one of the three case studies, that factor was included in the developed ESQM-V2.
3.7 Case Study Data Collection

Data Collection for case studies can rely on many sources (Yin 2009). Interviews, software metrics, observations were sources used for collecting evidence. As each case study differed the data collection method had to be implemented accordingly.

- **Embedded Software Quality Process** – The primary sources of evidence were software metrics. The software metrics were analysed and examined against the software metrics used in Software-MNC. Positive and negative trends were identified from studying the metrics.

- **Software Key Practices: Supporting merger and Acquisition** – Interviews were conducted which allowed for the embedded software development process to be studied. Software metrics for Small-Team were captured as part of the first customer release completed as part of Software-MNC. The combination of interviews and software metrics allowed for a comparison to Software-MNC.

- **FPGA For Telco Use Case** – Interviews and observations were used equally. In this case study I was a participant observer in this team. Having the role of Software Project Manager allowed me to complete observations for the duration of the project. Software metrics were captured as part of the two software releases made to customers. These were analysed against the software quality criteria outlined in Software-MNC.

3.7.1 Interviews

Interviews were conducted in both case study two and case study three. The questions in both case studies were open-ended allowing the interviewee to expand on their answer.

The aim of the interview in case study two was to assess the software process. A set of questions was devised from the key practice assessment already in
use at Software-MNC. Interviewees were selected based on their experience and the input they had in the software development process. Small –Team had three locations. A Software Engineering Manager and Development Lead were selected in each of the three locations. This ensured that to ensure consistency in comparing and evaluating the integration of Small-Team.

The aim of the interview in case study three was to understand the motivations of each of the project sponsors in the Marketing, Software and Hardware organisations. A set of six open-ended questions was devised. The interviewees were asked to elaborate or clarify any additional detail. Each person selected oversaw the development and activities and liaised with the senior management at a business and organisational level.

Three people were selected for interview,

- **Software Engineering Director** - Holds an electronic engineering degree and has twenty years experience in the telecoms industry. Oversees around twenty projects in the software division focused on communications.

- **Technical Marketing Engineer** – Holds a Computer Science and has fifteen years experience in the telecoms industry. Collaborates with customers to design solutions.

- **Hardware Engineering Director** - Holds an electronic engineering degree and has twenty years experience in hardware engineering. Managing a team of 45 in 4 different teams, design, pre-hardware verification, manufacturing and physical design.

### 3.7.2 Software Metrics

Software quality is determined by evaluating against a set of documented software metrics. Software Quality in all three case studies was evaluated using the set of metrics as documented by Software-MNC. Using a
standard set of metrics allowed for single comparison and measurement of software quality across all three case studies.

Software metrics were measured for each project in Software-MNC. There is a requirement for each project team to document and archive software metrics. These documents are stored in a database accessible to the division. The researcher was employed by Software-MNC and had full access to the software metrics for projects within the division. Documents were selected and examined based on their relevance to all three case studies.

3.7.3 Participant Observations

The researcher was the Software Project Manager on the FPGA Telco team. Observations allow researchers to check nonverbal expressions, determine who interacts with whom and grasp how participants interact with each other (Kawulich 2005). Observing the team allowed for a greater understanding in team dynamics. The FPGA Telco program required hardware, software and marketing cross collaboration in a GSD environment. For the hardware team this was the first time they were involved in a project that was less structured than typical projects. The hardware engineers on the team had experience in developing ASIC’s and chipsets for production products in the telecommunications industry. Observing hardware engineer to software engineer interactions solidified some of the observations seen in the literature review.

3.8 Conclusion

The research phases were designed so that any existing biases I as the researcher had were removed. The literature review was the first step in understanding what factors contributed to embedded systems and embedded software development in organisations outside of Software-MNC. While Software-MNC’s product portfolio and industry foothold is quite varied it was key for this research that a comprehensive literature review was
completed. This allowed for investigation to happen in a variety of companies who were different to Software-MNC. Start-ups and education initiatives were some of the case studies chosen from the literature that provided a contrasting view in how embedded systems are developed.

There is a limitation to this research as all three case studies were carried out in Software-MNC. Each case study investigated a certain component or relationship of the ESQM-V1. Each case study incorporated different methods. Software metrics were present in each of the case studies, software metrics allow for a measurement of software quality. Having the software quality component present in each case study allowed for an even comparison between case studies. Interviews and observations provided rich information to be gathered, as to the common stumbling blocks that arise during embedded systems development. It was through the study of the literature and case studies that the ESQM-V2 was developed.
Chapter 4 - Case Studies

Case one - Embedded Software Quality Process

4.1.1 Introduction

The case study looked at traditional software quality factors in Software-MNC that had been derived from standards and metrics ISO 9000 and CMMI. While there was input from hardware and marketing input discussed in the case study it was minimal and not seen as a key factor. The product in development followed an adopt-modify process for both hardware and software, components were reused and had been implemented in products prior to Comms-River and Comms-lake. Figure 4.1 illustrates how the software key area was the only area relevant in this case study, as only software metrics were evaluated.

Figure 4.1 Case Study One – ESQM-V1 Representation
4.1.2 Overview & Results

A software profile summary was conducted while the researcher was working with the software quality team in Software-MNC. Two profiles were completed on products for embedded telecommunications products. Product 1, Comms-River was released in 2011 and the software profile was completed in 2012. Product 2, Comms-Lake was released in 2013 the software profile was completed in 2014. Product 2 was a follow on from product 1. Comms-River was a “develop” program Comms-Lake was an “adopt-modify” program.

Alpha – Software and testing were 50% complete. Defects did not need to be fixed but no critical or customer impacting defects.

Beta – Software and testing were 80% complete. 40% of defects could remain open but no critical or customer impacting defects.

Gold – Software and testing were 100% complete. 10% of defects could remain open but no critical or customer impacting defects.

Comms-Lake Software Release:
- 4 Alpha
- 1 Beta
- 1 Gold

Comms-River Software Release:
- 1 Beta
- 1 Gold

There was an effort to reuse software from the Comms-River program in Comms-Lake. The categories that were analysed are:
• Code Review
• Static Code Analysis
• Code Churn
• Code Comment Ratio
• Software Defects
• Request for Requirement Change
• Customer Feedback

4.1.2.1 Code Review

Code Reviews should happen early and often to be most effective. In Comms-River there was an emphasis on formal reviews in the first release checkpoint. The number of formal reviews declined but there was a slight increase in the number of formal reviews ahead of the last release milestone. Walkthroughs were used as the main code review process ahead of 0.7 and 0.8. It was recommended by the software quality engineer that a review early, review often approach should be implemented in Comms-Lake, the chart is skewed to the right.

In Comms-Lake and Comms-River three types of code reviews were used. A formal review required the software developers to review the whole file. The developers involved in the review would meet to discuss the implementation having reviewed the code before the meeting. A walkthrough required the developers to review the code in their own time and approve the implementation. A code churn review would highlight any changes made to a file that had been previously reviewed in a formal review.
Code Reviews identify technical complexity, identify defects and allow developers to become familiar with all parts of the code (Singh & Singh 2013). The chart “Defects found peer review” demonstrates that there were more defects found the more code reviewed. Most of the code was reviewed before the final release. This did not allow the programme to understand technical complexity early in the program, which may have added additional risk in evaluating what should be prioritised from a software quality perspective.
4.1.2.2 Static Code Analysis

Static code analysis was used to only identify security threats. Comms-Lake and Comms-River both have a security component. The heartbleed bug was a serious vulnerability in the Open SSL cryptographic software library. A number of Linux distributions were affected which are commonly used in embedded systems (Codenomicon Defensics 2014). Embedded software can be more vulnerable to security threats as it may not be subject to physical security barriers such as badge access. Security issues are often difficult to find in complex embedded software. Security issues like heartbleed have implications for the organisation providing the software and
using the software. As an example, the Heartbleed highlighted how security threats can be introduced. In both programs there was a high percentage of false positives identified. The tools team recommended tuning the tool used in order to reduce the number of false positives and increase the accuracy of identifications. Issues identified as false positives in Comms-River were poorly commented. A recommendation was made to increase the quality of comments for identifications that were false positives or that the development team marked as “ignore”. There was a 10% increase in the number of identifications fixed.

Figure 4.6 Static Analyses for Comms-River & Comms-Lake

4.1.2.3 Code Churn

Code churn was measured on the number of lines of code that were added, modified or deleted. Code churn was only a valuable data point in Comms-Lake, data was analysed between four alpha milestones. Software quality outlined the following in Software-MNC:

- <10% code churn is a Low Risk
- 10% -20% code churn is Medium Risk
- <20% code churn is High Risk
Churn between milestones was in the medium-risk category. The overall churn of the program was in the high-risk category.

<table>
<thead>
<tr>
<th>STABILITY Attributes</th>
<th>Alpha 1 milestone</th>
<th>Alpha 2 milestone</th>
<th>Alpha 3 milestone</th>
<th>Alpha 4 milestone</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC Changes/Code churn</td>
<td>660.47</td>
<td>758.70</td>
<td>688.79</td>
<td>631.78</td>
</tr>
<tr>
<td>LOC Added</td>
<td>254.57</td>
<td>218.95</td>
<td>418.05</td>
<td>482.78</td>
</tr>
<tr>
<td>LOC Deleted</td>
<td>20.35</td>
<td>21.97</td>
<td>21.16</td>
<td>11.65</td>
</tr>
<tr>
<td>LOC Modified</td>
<td>42.37</td>
<td>34.71</td>
<td>28.31</td>
<td>22.52</td>
</tr>
</tbody>
</table>

Figure 4.7 Code Churn for Comms-River

<table>
<thead>
<tr>
<th>STABILITY Attributes</th>
<th>Alpha 1 versus Alpha 2</th>
<th>Alpha 2 versus Alpha 3</th>
<th>Alpha 3 versus Alpha 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC Changes/Code churn</td>
<td>252.889</td>
<td>294.359</td>
<td>347.797</td>
</tr>
<tr>
<td>LOC Added</td>
<td>205.916</td>
<td>203.974</td>
<td>202.924</td>
</tr>
<tr>
<td>LOC Deleted</td>
<td>30.02</td>
<td>28.02</td>
<td>27.02</td>
</tr>
<tr>
<td>LOC Modified</td>
<td>30.00</td>
<td>32.02</td>
<td>24.09</td>
</tr>
</tbody>
</table>

Figure 4.8 Code Churn for Comms-Lak

4.1.2.4 Code Comment Ratio

Software Quality in Software-MNC outlined the following:

- < 3 : 1 Ratio is rated as Low Risk
- 3 & < 6 : 1 is Medium Risk
- >6 : 1 is High Risk

Code comments are necessary to describe what the code is actually doing. They are beneficial to the customer to further help their understanding of the code, enables reuse of the code and future support should new developers join the project. In both programs this was a positive data point. Of the seven data points shown in figures 4.9 and 4.10, only one is in the high-risk category. The software quality team in Software-MNC required Gold Software to meet the medium risk criteria.
4.1.2.5 Software Defects

The later a defect is introduced in the software development lifecycle the more costly it is to fix. Investing time in finding defects early in the process is important. The ideal software defect chart would be skewed to the left. Comms-River shows that a lowest number of defects were reported in the first milestone. The second milestone saw an increase in the number of defects found which then gradually decreased. While there was a decrease in Comms-Lake from alpha to gold this was not a major decrease.
All software releases were released with defects. Details on each of the defects were included in release notes. The decision to release without fixing these defects was owned by the software director. Input from other stakeholders was taken from marketing, software quality and senior engineers. The hardware team was able to log hardware defects that could potentially impact software development. For both programs over 50% of these defects were rejected. A recommendation was made by the software quality engineer for software to triage these defects earlier to fully assess the
impact to software. This recommendation was not followed through for the Comms-Lake Program.

Figure 4.13 Released Defects Comms-River

Figure 4.14 Released Defects Comms-Lake
4.1.2.6 Request for Requirements to Change

There are two classifications when requesting requirement changes.

1. If the request does not have a major impact on scope, schedule or budget, this will be referenced as CR-1
2. If the request does have a major impact on scope, schedule or budget, this will be referenced as CR-2

Both classifications were treated as technical debt and an indication of changing requirements. Requests originated from:

- Customers,
- Architects
- Technical Marketing Engineers
- Software Engineers
- Inherited from older projects

CR-1’s give an insight into the smaller requirements change examples including implementation of functionality, platform configuration and how data is passed and dealt within the software. There was a significant decrease in the number of CR-1’s that were logged in Comms-River from Comms-Lake. A total of 351 were recorded in the Comms-Lake program. 20% of CR-1’s recorded were “rejected” by the program team. 80% of CR-1’s recorded were recorded and managed before the “beta” milestone. This indicated that there was pro-active management of requirements change in the program. There was an increase in the number of CR-1’s from the “beta” to “gold” milestone. An investigation should be completed as to why there was an increase in the requirements before the “gold” release was completed.

This was significantly reduced to 20 CR-1’s in the Comms-River Program, 40% were “rejected”. 30% were recorded before the final release. Both programs had a similar number of resources assigned. Comparison of percentages in the programs is misleading, as this does not reflect that there
was a stark reduction in CR-1’s recorded and there were less than ten that needed to be managed in the “gold” release of Comms-Lake.

CR-2’s give an insight into some of the major requirement changes examples include hardware changes and removal or addition of software functionality. There were over 170 CR-2’s recorded in Comms-River:

- 105 were hardware changes that could impact software
- 67 were submitted by the software team

80% of CR-2’s were implemented. Further analysis should be done on the impact of accepting these requirement changes.

4.1.2.7 Customer Feedback

The software quality engineer made a recommendation to survey the customers who had received early access software as part of the Comms-Lake program. This was not common practice in Software-MNC. The objective of the survey was to understand the customer’s perspective in terms of the software quality. As Comms-River was an “adopt-modify” program it was decided not to survey customers.

The questions asked were:

- Early Access Software is provided early enough in our design validation phase
- Feedback of Early Access software is acted on in a timely manner
- Early Access software contains the necessary functionality/features making our early validation and design work valuable
- Early Access software is of desired quality and reliability
- Early Access software is accompanied with the necessary technical support
- Early Access software has helpful documentation

Over 80% of the respondents agreed. One of the respondents rated a neutral on 4 questions and 2 disagree for two survey questions:
• Early Access software is accompanied with the necessary technical support
• Early Access software has helpful documentation

Figure 4.15 Customer Respondents Comms-River

4.1.3 Summary

Both Comms-Lake and Comms-River were measured equally. The software quality targets as defined by Software-MNC were met in both programs. Both programs accepted ~80% change in overall requirements. In Comms-River 60% of CR-2’s were directly impacted by changes in hardware. The customer survey was mostly positive however none of the respondents selected “Strongly Agree”. Technical Support and Documentation were highlighted as areas for improvement. While the software quality criteria were reached for both programs, the team could have invested more effort in ensuring that there was a focus on customer satisfaction. As highlighted in the literature 2.3.2, embedded software developers often neglect goals and constraints that would normally not be neglected. Recommendations from the software quality engineer were
provided to the team after the Comms-River program. Comms-Lake showed an improvement in some of the metrics. The decrease in changing requirements in the Comms-Lake program was minimal. While software standards provide a foundation for creating internal software metrics, external factors need to be included so that the customer focus can be at the core of the program.

4.1.3.1 Inputs into the ESQM-V2

In both programs changing requirements impacted on the perceived quality and value of the product. The changing requirements impacted the schedule and did not allow the new features to be co-designed by the hardware and software teams or a complete requirements definition to be scoped. While activities like code review and defect count are necessary activities in the embedded software development process they were viewed as sub activities and should be measured and completed in all embedded software development programs. Software-MNC completed two activities to gather customer feedback. The first was releasing an early access version of the software, which enabled a customer feedback loop. The second was to complete a survey. The focus was only put on the customer later in the development cycle. A number of the requirements changes originated from the customer. Some of these requirement changes could have been avoided if customer focus had been completed earlier in the development cycle. Customer focus needs happen early in the development process. A feedback look needs to be built in so that the right product can be designed for customers.
Case Two - Software Key Practices: Supporting merger and Acquisition

4.2.1 Introduction

Small-Team was an acquisition by Software-MNC. The researcher investigated how Small-Team implemented software quality factors based on the same standards used in Software-MNC. Development and strength areas were outlined for both Software-MNC and Small-Team. Through the software analysis, potential impacts to the embedded software development process in both Software-MNC and Small-Team could be quantified. Figure 4.16 illustrates the key areas that were investigated. Software quality profiles were analysed as part of software releases completed in Small-Team under the software quality criteria key area. When Software-MNC acquired Small-Team this was a business and organisational change for both companies.

Figure 4.16 Case Study Two Embedded Software Quality Framework
4.2.1.1 Overview

The study was carried out during a merger of two teams, which resulted from the acquisition of a small company by a multinational. Software-MNC, is based in Ireland. The small company, Small-Team has teams in three locations - America, The UK and China. The evaluated teams both develop embedded software products. Small-Team had been acquired twice before. This chapter introduces the companies and presents information on software process key practice assessments that were completed in both Software-MNC and Small-Team before integration of the teams had begun.

4.2.2 Software Key Practice Assessments within Software-MNC

Software-MNC use an internal “Software Key Practice Assessment” which assesses seventeen key practices devised from the Capability Maturity Model, CMMI (Paulk et al. 1993). Using this assessment, the software quality team regularly conducts evaluations on software teams in the company awarding a score between one and five based from the specified criteria. Table 4 outlines the results of the key practice assessment completed in Small-Team.
4.2.3 Internal Assessments as an Integration Framework

The key practice assessment allowed Small-Team to describe what software development practices were important to their team when developing software. As Small-Team will have pre-existing customer commitments it is important that these can be met during the transition period. It also gives an opportunity to Small-Team to build trust in Software-MNC. As outlined previously acquisitions are bought to add value. Although Small-Team will most likely be fully integrated into Software-MNC, the key practice assessment it provides a framework to listen to Small-Team and
understand their process before any major changes are made. Any changes that are made can then be made together improving the strengths of both organisations. It also gave the opportunity to change practices that staff in Small-Team did not feel were working and could be improved. Software-MNC can offer help to change existing processes. Change and integration can then come from Small-Team.

Three areas were identified that Software-MNC could acquire better methods during software development.

- **Code Review** – All code was reviewed on check-in
- **Defects** – Software released with zero or a small number of minor defects
- **Document Creation and Management** – All documents were reviewed and discussed in depth

### 4.2.4 Prioritising Software Development Practices

Requirements, comprising of three of the key practices and estimations, were identified as focus areas during the integration process. The three teams operate as a global team working towards the one objective, as the literature has shown control of the project often still remains in a US team. Using the existing key practice assessment identified key practice areas that Small-Team would benefit from additional training and coaching from expert teams in Software-MNC to better integrate. It also highlighted areas for improvement directly related to GSD that were in existence before the acquisition. As discussed, earlier communication and coordination are more difficult to maintain in GSD teams. Control for most GSD projects remains in the team located in the US. One of the interview candidates outlined that there was little contact between the key decision makers in the US. The four key practices that were identified were controlled by the team in the U.S. where most of the key decision makers were located.
4.2.5 Learning from Small Team

Software-MNC will want to integrate Small-Team so that they are aligned. It is important that Software-MNC learn from the processes that have made Small-Team the successful. Code review, defect management and regular code commits were identified as strength areas. The data collected from the interviews was compared to the internal assessment data presented on table 4. Defect management and code reviews were given a low score. Defects reported from the products released by Small-Team were under five, the average number of defects reported for software releases from Software-MNC was over twenty.

4.2.6 Business and Organisational Impact to Small-Team

Small-Team had a closer and more collaborative relationship with the customer than studied in the first case study. Engineers in Small-Team fulfilled some of the marketing factors that are outlined in the ESQM-V1, which were not observed in Software-MNC, such as:

- Customer Engagement in the Design Process
- Rapid Prototyping
- Customer Collaboration

Engineering teams talked directly to the customer throughout the software development process and provided technical guidance and debugging when needed, often releasing code on a daily basis. This interaction meant the customers had a positive experience. There were a number of activities like code review and documentation that are necessary when guaranteeing software quality assurance.

The acquisition by Software-MNC meant a change in how Small-Team supported the customer. Software-MNC has a well-defined release process. Small-Team had to adjust to this release process and change the customer support model. Customers expressed their concerns that if their daily
interaction with engineers was reduced that this would impact on the value of
the software delivered by Small-Team.

Employees in Small-Team discussed how M&A’s were either positive or
negative. Negative, where a software development processes was imposed by
the dominant entity resulting in strained relationships for all teams involved.
An M&A was beneficial for both entities when both software development
processes were “joined whole”, source quotation from Small-Team
employee.

4.2.6.1 Inputs into the ESQM

Organisations who are completing an acquisition need to be open to
change as much as the organisation being acquired. Small-Team always had
customer focus at the centre of their development process. This resulted in a
product that satisfied the customer. The team focused on the design and
ensuring that all team members were agreed. Prototyping was used to good
effect allowing for design problems to be anticipated or innovative ways of
implementing new features to be trailed. The overall outcome of most of the
projects was a code base, which could be reused and adapted, lowering the
design effort for future products.
Case Three - FPGA For Telco Use Case

4.3.1 Introduction

This project was a proof of concept designed to demonstrate split band baseband processing. This was in a market segment that Software-MNC does not have a strong market share in but one that is strategic for business development. A team of ten comprised of hardware and software engineers, technical marketing and project management. While the project was inline with organisational strategy and seeks to grow market share, this was not examined in the case study. The business and organisational component did not have a significant impact in determining project success. Figure 4.17 illustrates the key areas that were investigated. Software quality metrics were evaluated as part of two software releases. The program was funded by Software-MNC in response to the changing telecommunications industry. Hardware engineering, marketing and software engineering were all represented on the FPGA telco team.

![Software Quality Model Representation](image)

Figure 4.17 Case Study Three Software Quality Model Representation
4.3.1.2 Program Overview

TelcoFPGA was a program to demonstrate new architectures in baseband processing. The aim of the program was to provide a reference platform to operators and telecommunications equipment providers. The team was dispersed over four locations: Ireland, U.S., Canada and three offices in China.

The program was run in 2014 for a planned duration of six months. There were two delivery cycles planned. The first was a software only release: this was a version of the software that could emulate some of the FPGA functionality. The final customer delivery included the FPGA and a software release.

Delivery 1:

1. 0.1.0 Software Only
2. Open Source Software Version 1.7
3. Real-time Operating System Version 5
4. Server Platform with Sandy Bridge Architecture

Delivery 2:

5. 1.0.0 Software
6. Open Source Software Version 1.8
7. Real-time Operating System Version 6
8. Server Platform with Haswell Architecture
9. FPGA
The software engineering team did not have to develop or contribute to any of the open source software projects. Integration was required with the software stack and sample application that was developed by the software engineers on the team. The software engineering team had to provide Meta layers in the form of board support packages to enable the third party operating system provider, to deliver an operating system that would meet the real-time requirements and virtualization requirements. The third party operating system provider was viewed as an eco-system partner.

4.3.1.2 Team Structure and Distribution

The program manager was responsible for overseeing all deliveries in the program. The software program manager and the hardware architect were responsible for the respective deliveries. Interaction with the third party operating system vendor was viewed as an eco-system partner. This engagement was co-owned by the software project manager and a marketing representative. Table 5 outlines the program team, stakeholders, location and the level of involvement/input.

4.3.2.3 Meeting Structure

Core team meetings were run weekly, updates to stakeholders were scheduled monthly. Eco-System partner meetings were scheduled twice monthly. Daily integration meetings were setup for the third integration milestone. Ad hoc meetings were setup as required internally for the various functional teams or cross – functional teams.
<table>
<thead>
<tr>
<th>Core Team</th>
<th>Ireland</th>
<th>US</th>
<th>Canada (Eco-System Partner)</th>
<th>China 1</th>
<th>China 2</th>
<th>China 3 (Eco-System Partner)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Manager *</td>
<td></td>
<td></td>
<td></td>
<td>SW Lead *</td>
<td></td>
<td>OS Engineer *</td>
</tr>
<tr>
<td>Software Marketing *</td>
<td></td>
<td></td>
<td></td>
<td>Software Quality*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Project Manager *</td>
<td></td>
<td></td>
<td></td>
<td>SW Engineer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware Architect *</td>
<td></td>
<td></td>
<td></td>
<td>SW Engineer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTL Engineer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pre Hardware Verification</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Engineer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW Lead *</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW Engineer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SW Engineer VT &amp; OS *</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform Application Engineer *</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Technical Marketing Engineer *</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consultancy</td>
<td>Wireless SW Architect</td>
<td>Real-time OS</td>
<td>Wireless SW Architect</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder</td>
<td>SW Engineering Manager</td>
<td>Wireless Marketing Director</td>
<td>Engineering Director</td>
<td>SW Engineering Manager</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hardware Engineering Director</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software Engineering Director</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Assigned to the program part-time
4.3.2.4 Software Quality Criteria

The scope of the program was to provide reference software. The software had to achieve “Proof of Concept” quality criteria. The software had to reach the following criteria before release approval was granted.

- Code Review
- Static Analysis
- Code Coverage
- Passing Tests
- Customer Documentation
- Defects Tracked
- Requirements Document
- Software Licensing Review
- Virus Scan

4.3.3 The Iron Triangle and Project Success

Time, cost and scope remain project success factors. All interviewees oversee multiple projects and customer engagements. Projects need to be planned per year and reports given to senior management. It was important for all interviewees that the resources required for project success could be estimated and understood. Customer Feedback and collaboration was a key factor for the software engineering director when evaluating project success, a customer gives a project ‘a sense of reality’. Customer collaboration gives you perspective on what you thought the customer wanted versus what they actually wanted. The software engineering director recognised the importance of customer collaboration when developing prototype products. While the core components of the iron triangle were important in measuring project success for the software engineering director and technical marketer viewed customer collaboration as a success factor.

4.3.4 Software Quality – ‘in simple terms is just pure metrics’

Software quality can be measured. It “in simple terms is just pure metrics”; all three interviews stated that software should meet customer
expectations. For the software engineering director technical debt was a concern, which needed to be factored in throughout the development process. “You need to make sure that you are not leaving things behind for the next project”.

More recently there has been a shift when designing hardware. Traditionally software was 6 to 9 months behind the hardware. There have been programs where software development was ahead of the hardware development. Pre-hardware validation, emulation and simulation are important to verify content but it is insufficient “I view software as an extension of the pre-hardware validation team”. Software is a gating factor for one of the major RTL milestones. It introduces more dependencies in the program but it is “worthwhile in terms of quality”.

4.3.5 Conflicting Opinions

The Technical Marketing engineer was used to research and development path-finding model. Engaging with customers to develop and design new solutions for markets that are only in development. The Hardware Engineering Director found it difficult to switch to a path-finding model from an environment where hard requirements are documented before a project is started. The biggest gap in terms of project objective existed between the Hardware Engineering Director and the Technical Marketing Engineer. The lack of clear requirements made it difficult for the Hardware Engineering Director to commit resources when the vision understood my marketing was not clear to him.

4.3.5.1 Inputs to the ESQ Model

Software was no longer seen as following the hardware development process. This was the first time during the research that software was seen as parity with hardware when developing embedded systems. In embedded systems development, both components should be delivered together.
Roadmaps should align to ensure that hardware is used to test software and vice versa.

More effort should have been invested in communicating the objective of the program to all project stakeholders. There was a clear gap between marketing and hardware, this did not appear to be a problem in software. However the Software engineering director would have liked earlier customer collaboration to validate the requirements that marketing were requesting from the software teams.

Software in embedded systems is not a “throw-away” component. Software metrics need to be used to measure the overall software health. Hardware leverages software as a test bench. Marketing teams view software capabilities as a way in which to engage with customers when selling solutions. Through this case study it became apparent that customer focus needed to be one of the key factors in ESQM-V2. While team members may not be in the same division or country, it is important that the team be as unified as possible where all feel ownership in the project.
Chapter 5 - Findings

5.1 Overview

The literature review and the case studies highlighted that there are multiple factors that impact on the quality of embedded systems and embedded software quality. There are four key functional areas that have a direct impact on embedded software quality:

- Hardware
- Marketing
- Business & Organisation
- Industry Changes

Each factor in one of the four key areas needs to be managed during the product development cycle. The factors identified show where the functional areas need to be aligned to ensure embedded software quality.

5.2 Relationships and Interactions amongst the four key factors

All four key areas impact embedded software quality directly or indirectly. Figure 5.1 shows how the four key areas and software interact. Industry Changes impacts the Business & Organisation. It is Marketing’s role to ensure that the Business and Organisational strategy is realised. Marketing also needs to look outward constantly incorporating Industry Changes. Marketing provide the main interface from the Industry and Business to Hardware and Software teams when designing a product. The literature reviewed maintained how software traditionally follows hardware schedules. However there is a change in the role software plays in embedded systems. In the FPGA Telco case study it was stated that software is a gating factor in hardware development. Embedded software is fulfilling functions and features that were traditionally implemented in hardware or mechanics, machine learning and artificial intelligence.
The relationship between factors needs to be understood at a high level as in figure 5.1 so that relationships between lower level factors as shown in the ESQM-V2 can be understood in greater depth.

5.1.2 Updating The Initial ESQM-V1

The ESQM-V1 showed how activities in each the key areas could impact on embedded software quality. The literature highlighted how embedded systems development happens largely in isolation with each function interacting only when necessary. ESQM-V2 focused on understanding essential activities that need to occur when developing embedded systems across each of the key areas. In order for a project to be successful each of the ten factors need to have an environment of trust respect, team members need to ensure that communication and coordination are not forgotten about.

**Embedded Software Quality Process** - The software profile analysis completed for Comms-Lake and Comms-River showed how the customer can have a significant input in requirements. An embedded software team
needs to have the capacity to deal with additional requirement requests from customers. Software metrics are essential in looking at the quality of the software. They allow for an assessment to be made for software reuse and maintainability.

**Software Key Practices: Supporting merger and Acquisition** – Small-Team had a strong focus on customer collaboration, which was not as present in Software-MNC. Due to the customer collaboration there was a feedback loop in existence that easily allowed for changing requirements to be introduced into the embedded software development process. Hardware and Software engineering acted like a single team. This was visible when document reviews were completed. All team members met reviewed and agreed project documents before development was started. There was focus on ensuring the design was correct which resulted in a product that had fewer defects and satisfied the customer. Prototyping was used to good effect; this work was shared with the customer. As the IP in use was of a high quality and commercialised in several products it was easier to reuse and maintain.

**FPGA For Telco Use Case** – The Hardware and Software Engineering Directors would have liked to have seen customer interaction much earlier in the process. A disconnect existed between hardware and marketing. Weekly meetings focused mostly on engineering and status updates. Often the technical marketing engineer who was closest to the customer was absent. This reduced the teams understanding of what the customer wanted. Unlike Small-Team there was a clear divide in roles from each of the functions. Separate documents were maintained for each of the functions none of which were reviewed as a team. The design had to change through the program resulting in a lower level of software quality compared to Small-Team.

When developing embedded systems and embedded software working as one team is vital to quality of the overall product. The design of the product
needs input from the whole team. The link between the customer is essential to deliver the right product and to be able to change it if the customers requirements have changed.

5.2.1 Complex Design

![Figure 5.2 Complex Design Key Area Interaction](image)

Industry Changes **INFLUENCES** Marketing

Marketing **INFLUENCES** Hardware & Software

Hardware **MUST CO-DESIGN** Software

Embedded systems and embedded software are increasing in complexity but need to be delivered in a shorter timeframe (Theelen et al. 2007; Sgroi et al. 2000; Oliveira et al. 2008). The design must also incorporate customer requirements and industry standards. A reliable system must be delivered at the end of the product development lifecycle. Embedded systems are replacing mechanical systems as they provide a higher level of functionality (Date & It 2014). Products, which traditionally had no engineering complexity or connectivity, will soon become intelligent. The customer is demanding smarter devices in a more connected world.
As the number of SLOC increases so does the embedded software complexity. Hardware and Software need to co-design to design a reliable solution. There are unique constraints that both engineering functions need to consider. Trade-offs will often exist in developing a system that is reliable.

5.2.3 Customer Focus

Customer strategy is set at the top level of the organisation and at a business group level. It is marketing’s role to deliver on the customer strategy and ensure:

1. Customer Satisfaction
2. Customer Experience
3. Customer Requirements are incorporated
4. Customer Benefit

Not only should the organisation set strategy but also ensure structures are in place to enable the marketing team, legal, IT infrastructure, etc… Many
technology companies recognise the value that suppliers give to the organisation through supplier awards. Vodafone is just one company that recognises its suppliers in several categories: outstanding delivery, responsible supplier award and outstanding commercial value and product development award (Vodafone 2015). For companies awarded a supplier award it demonstrates that the company creates value and the customer wins by choosing that brand (Freeman et al. 2004).

It is the role of marketing to interact with the customer. A holistic view of customer requirements is required. In embedded systems this can include feature requests, expected quality, performance and an understanding of the customers market and strategy. Marketing should facilitate collaboration with engineering so that the customer should be involved in the early design process so that the customer gets the best solution.

5.2.4 Risk

![Figure 5.5 Risk Key Area Interactions](image)

Business & Organisational **INFLUENCES** Marketing

All products developed will have to deal and mitigate against a certain level of risk. Risks need to be mitigated against at an organisational level and a customer level. A business will structure its organisation so that it will deliver customer and shareholder value but allow it to take risks while ensuring that their sufficient fall-back plans are in place. This can be done through global teaming, mergers and acquisitions and outsourcing. While a strategy will aim to reduce risk, managers need to fully assess how it will impact the overall organisation (Freeman et al. 2004).
A customer accepts a certain level of risk when they choose a supplier (Sargut & McGrath 2011). Marketing must understand the organisations strategy to mitigate risk and also risk mitigation for the customer. Embedded systems are part of a chain. Marketing can mitigate risk for a customer by delivering the right product on time inline with customer expectations.

5.2.5 Industry Standards

![Diagram of Industry Standards Key Area Interaction]

Industry Changes **INFLUENCES** Hardware & Software

Hardware **MUST CO-DESIGN WITH** Software

Hardware and software engineering teams need to design a product for the customer. However this must comply with Industry Standards. There are strict regulations in the automotive, healthcare and telecommunications industries.

For example European Telecommunications Standards Institute (ETSI) telecommunications, Federal Aviation Authority (FAA) aviation, National Standards Association Authority (NSAI) healthcare and the International Standards Organisation (ISO) are just some of the standards bodies that govern how a product should be designed. The governing standards body will implement standards to cover all areas in the industry. How the electronic device is designed, how data and privacy are protected. As the industry changes they will look to define new standards to be used in the industry.
Engineers cannot ignore the respective standard. An in depth knowledge of the standard is required by both hardware and software teams. Embedded systems are part of wider system hardware and software engineers need to co-design a system in order to ensure that the standards are met. Software needs to understand how hardware will implement features so the engineers can ensure that the project will not fail because standards were not adhered to. As there are strict performance requirements and power consumption embedded software engineers need to ensure that the software designed is not computationally expensive in terms of its power consumption when the software is running.

5.2.6 Abstraction & User Interfaces

![Figure 5.7 Abstraction & User Interface Key Area Interaction](image)

Abstractness is the capacity of the system or component to be extended (Oliveira et al. 2008). Hardware and software components should be designed with abstraction in mind for an embedded system. Blocks and components that are not designed with abstraction in mind are hard to reuse. Defect count can be reduced in embedded systems by reusing components. Extensible design will facilitate the exploration of different architectural solutions. It is necessary to separate certain components like communication and compute (Sgroi et al. 2000).
User interfaces are the links that enable components and blocks to communicate. Interfaces will be defined internally to both software and hardware and externally so that software and hardware can communicate. Hardware and software engineers need to understand what variables are being passed and their formats. Industry standard interfaces need to be understood and implemented. Embedded systems should limit the use of different interfaces.

A high level of abstraction and standardising user interfaces results in reducing complexity. It facilitates prototyping as components can be used like building blocks.

5.2.7 Prototyping

![Figure 5.8 Prototyping Key Area Interactions](image)

**Figure 5.8 Prototyping Key Area Interactions**

Industry Changes **INFLUENCES** Hardware & Software

Hardware **MUST CO-DESIGN WITH** Software

The role of a prototype should be determined before prototyping begins. Prototyping should aid determining the best architecture and facilitate requirements definition. Engineers will have a clearer understanding of how each block will interact. Mapping of functional architecture is an essential step in understanding the configurable system (Sgroi et al. 2000; Sangiovanni-Vincentelli & Martin 2001; Steiner & Athanas 2005).

Marketing should ensure that the customer’s requirements are incorporated as early as possible in the process. Marketers often believe that they are not
Engaged enough in the early design process. (Fisk, 2004) Engaging customers early in the design process reduces risk as a more collaborative approach the customer can be satisfied early in the design process that the product being developed is the product they expect. Boeing engaged eight airlines when designing their Boeing 777 (Sargut & McGrath 2011). As recorded in 2015 the 777 has more orders than any other jet on offer.

5.2.8 Requirements Definition

![Figure 5.9 Requirements Definition Key Area Interaction](image)

Industry Changes INFLUENCES Hardware & Software

Hardware MUST CO-DESIGN WITH Software

Given the complexity of embedded systems clear requirements are an important factor during the product development cycle. The market does not know what to wish for (Karlsson et al. 2007). Clear understandable requirements are essential to align all project stakeholders. A requirement is anything that needs to be in the product. IEEE STD 830 (Committee 1998) outlines a requirement should be

1. Correct
2. Unambiguous
3. Complete
4. Consistent
5. Ranked for importance and or stability
6. Verifiable
7. Modifiable
8. Traceable

Poor requirement definitions can have a negative impact on the programme. They can be extremely costly in terms of misspent engineering resources or not meeting customer expectations. Requirements form part of the technical documentation in a program. Types of requirements should include functional and non-functional. Non-functional requirements can include industry standards to be adhered to, power consumption and performance. Functional requirements are often specific to what the software should do. Marketing should play a role in prioritising requirements based on their knowledge of what the industry and customer requires (Clements & Northrop 2001). It is inevitable that requirements will change in a program. Software and Hardware should ensure that a requirement is detailed sufficiently for it to be understood and implemented as described.

5.2.9 Responding to Changing Requirements

![Figure 5.10 Responding to Changing Requirements Key Area Interaction](image)

Hardware **MUST CO-DESIGN WITH** Software

There is a constant flow of changing requirements from customers, users and developers. There is no one reason why requirements change (Karlsson et al. 2007). An embedded system must be designed in a way that it can facilitate some level of changing requirements throughout the entire product lifecycle. A program can deal with changing requirements in a more efficient way if a clean design is in place and user interfaces and abstraction have been thought out. Both of these factors allow a certain level of flexibility (Huang et al. 2012; Drechsler & Breiter 2004). Changes in one component can affect other components (Ard et al. 2014); hardware and software must work together to
understand how changing a requirement will impact technically and to cost and schedule.

5.2.10 Maintainability & Reusability

![Figure 5.11 Maintainability & Reusability Key Area Interaction](image)

Hardware MUST CO-DESIGN WITH Software

Embedded Systems development is costly: many programmes follow adopt modify methodology. Component reuse is the goal of most IP methodologies (Sangiovanni-Vincentelli & Martin 2001; Ard et al. 2014). It reduces the cost, reduces development time and limits redesign. Hardware and software engineers need to ensure that maintainability and reusability is factored into the design process. Given the tight coupling in the areas, maintainability and reusability for the whole system needs to be incorporated.

During development in both disciplines is important that blocks or components are not developed in isolation. Hardware and software should not be developed in isolation, as integration is more complex and debugging time increases, as both teams must learn how each other’s component works. Ensuring that your system and software is maintainable and reusable will help facilitate changing requirements (Huang et al. 2012).

Code reviews and comprehensive documentation are important activities when designing code that is maintainable and reusable. They reduce overall defect count and allows for developers and engineers to understand the inner workings of the code even if they were not involved in the original implementation.

5.2.11 Testing
Figure 5.12 Testing Key Area Interactions

Hardware MUST CO-DESIGN WITH Software

Testing is used to verify and validate design correctness (Sangiovanni-Vincentelli & Martin 2001). Testing benefits maintainability and reuse; engineers can design tests that demonstrate real world scenarios and deliver systems with a level of confidence. Hardware and software teams will aim to find defects as early as possible in the development process. Individual Test strategies should be devised for hardware pre and post hardware availability and testing can include emulation and simulation. In software component level and integration testing plans should be in place. System test plans should be devised by both hardware and software.

Figure 5.13 Cost Per Defect in Embedded Software (Date & It 2014)

5.2.12 Respect, Trust, Communication and Coordination

Embedded systems development requires a team of varied skills. Each group must respect the other disciplines that are involved in the team. There must be a level of trust for the work that each team member needs to do. It is the norm for embedded systems development to occur in a GSD environment. There are additional complexities introduced when Embedded
Systems products are produced in a GSD environment. Communication and coordination become harder in a GSD and M&A environment. This makes trust and respect much harder to achieve. The ESQM-V2 does not outline how trust, respect and communication and coordination should be achieved only that they need to be considered.

5.3 Developed ESQM-V2

Using the ESQM-V1 derived from the literature review common factors were identified amongst the four key areas and software. Any factor that existed in 2 or more key areas is considered by the researcher to be a key factor in embedded software quality. Although the factors are common the factors may have a different function. Each factor was examined under the area it existed in and how each area needed to understand the dependency to other factors and to the other areas it was identified.

1. Impact – A marked effect or influence
2. Influence – The capacity to have an effect on the character, development or behaviour of someone or something, or the effect itself.
3. Facilitates – Make (an action or process) easy or easier

Figure 5.14 shows the interdependency of the 10 factors from each of the key areas. Table 6 describes the input required from each of the key areas for each of the 10 factors. The 10 factors were selected as they appeared most often in the key areas. It was necessary to complete grouping for some of the factors. The aim of the ESQM-V2 was to demonstrate how cross functional embedded systems development should be completed. Some factors from ESQM-V1 have not been included. The researcher chose to only investigate the factors in embedded systems development which were cross functional.

The return for an embedded systems team should they implement the ESQM-V2 would be embedded software that is valuable and of the desired quality
appropriate for the given project. The embedded software developed would be seen as a differentiator.
Figure 5.14 Embedded Software Factor Interaction Model
Table 6 Embedded Software Quality and Value Model

<table>
<thead>
<tr>
<th>Complex Design</th>
<th>Introduce new technology</th>
<th>Understand design sufficiently to communicate with Customers and set engineering direction</th>
<th>Implement new technology and architectures. Work with Software to design solution for current and future products</th>
<th>Implement new technology and architectures. Work with Hardware to design solution for current and future products</th>
</tr>
</thead>
<tbody>
<tr>
<td>Industry Standards</td>
<td>Define Standards</td>
<td>Adhere and Implement Industry Standards</td>
<td>SW Components are designed so that future reuse is possible</td>
<td>SW Components are designed so that future reuse is possible</td>
</tr>
<tr>
<td>Maintainability &amp; Reusability</td>
<td></td>
<td>Ensure components can be maintained and reused for new products</td>
<td>Ensure components can be maintained and reused for new products</td>
<td>Ensure components can be maintained and reused for new products</td>
</tr>
<tr>
<td>Abstraction</td>
<td>HW Blocks are designed so that future reuse is possible</td>
<td>Use input from Marketing to fully define requirements to be delivered by HW</td>
<td>Use input from Marketing to fully define requirements to be delivered by SW</td>
<td>Use input from Marketing to fully define requirements to be delivered by SW</td>
</tr>
<tr>
<td>Requirements Definition</td>
<td></td>
<td>Ability to change project scope based on new requirements during project development</td>
<td>Ability to change project scope based on new requirements during project development</td>
<td>Ability to change project scope based on new requirements during project development</td>
</tr>
<tr>
<td>Responding to Changing Requirements</td>
<td></td>
<td>Ability to change project scope based on new requirements during project development</td>
<td>Ability to change project scope based on new requirements during project development</td>
<td>Ability to change project scope based on new requirements during project development</td>
</tr>
<tr>
<td>Risk</td>
<td>Understand Risk to the entire Organisation</td>
<td>Understand Project Risks and Mitigate customer risks</td>
<td>Test against specifications and ensure sufficient testing completed before release</td>
<td>Test against specifications and ensure sufficient testing completed before release</td>
</tr>
<tr>
<td>Testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
“The evolution of products into intelligent, connected devices- which are increasingly embedded in broader systems-is radically reshaping companies and competition”

(Porter & Heppelmann 2014)

6.1 Summary – Answers to the Research Questions

The IOT market is growing at an exponential rate. The demand from the market for IOT devices means that embedded systems and embedded software face new technological and organisational challenges. In order for organisations to meet demand and customer expectations they must be able produce a high quality product in fast paced market. Embedded Software is a key component and differentiator when delivering innovative IOT solutions to customers.

The way a company chooses to use embedded software can enhance or damage their reputation. Tesla learned that under certain conditions there was a risk to driver safety. The company choose to push software updates to improve customer satisfaction. (Porter & Heppelmann 2014) Volkswagen developed software to detect when the car was under testing conditions and lower vehicle emissions during the test. (Economist 2015)

The Model T was the first mass produced car; there was not any software in the car. While Volkswagen and Tesla are both in the automotive industry both companies require embedded software in their design process. The quality value and use of software in a product must be carefully considered by the organisation.

Following the completion of my literature review, three research questions were identified. The research questions were designed to understand what
teams and factors outside of the software engineering organisation contributed to the overall embedded software quality. The term software quality has become linked with metrics. As an industry based practitioner my motivation was to understand the factors and activities that guaranteed value in the embedded software development process.

**What are the key factors that define success in the context of embedded software quality?**

Chapter 5 outlines ten factors that are essential in delivering embedded software quality. The factors have been derived from the literature review and the case studies. The developed conceptual model illustrates how each key area must provide input for the factors they influence. These factors need input from the whole embedded systems team.

- Complex Design
- Customer Focus
- Risk
- Industry Standards
- Abstraction & User Interfaces
- Prototyping
- Requirements Definition
- Responding to Changing Requirements
- Maintainability & Reusability
- Testing

**Who are the stakeholders involved in embedded system design?**

The initial ESQM-V1 illustrates how hardware, software and marketing functions are required when developing embedded systems. Embedded system development needs to be supported by the business and organisation. All projects should facilitate meeting business objectives. Many
embedded systems are used in highly regulated industries such as automotive, healthcare and aviation devices. Industries are highly regulated and there is strict regulation. Companies must pay close attention to changes in industry standards. The IOT is one industry that is transforming the embedded systems market, companies must look outwards to realise the potential market opportunities and not loose their market share.

*What is the role of teams who are fundamental in embedded systems design in ensuring embedded software quality and value?*

The software and hardware engineering organisations are responsible for delivering their respective components in an embedded system. It is the business and the marketing organisations roles to ensure that the right product is developed for the industry and the customer. The software engineering organisation cannot deliver embedded software that is of high quality and value without input and co-design from all areas in the company. A valuable product can only be developed if the goals and objectives are clearly understood by all team members. (Rong et al. 2011)

**6.2 Future Research**

There are limitations to this work, as the developed conceptual model has not been implemented. I would recommend the following:

- Examining the validity of the ten factors identified in the ESQM-V2 through action research that would include all of the five key areas.
- The relationship model is primarily based on qualitative analysis observe through the case studies. The developed conceptual model could be improved further to clearly guide practitioners on the best practices that result in embedded software quality, which is of high value.
- This research did not consider the use of formal system test in ensuring a validated and verified product.
6.3 Conclusion

The success of a product requires all stakeholders to be invested and believe in the value the product brings to the industry. Embedded software is no longer just an “add-on” it is key to product success and business survival. An embedded system cannot be designed by hardware teams, software teams or marketing teams in isolation the development process must be in unison to ensure that the best product is produced and the software in the product is of high quality, which in turn delivers value for both the customer and the business.
Appendix

Appendix I

Software Key Practices: Supporting merger and Acquisition Case Study One
Interview Questions:

1. Describe the requirements engineering process?
2. How are requirements defined and captured?
3. What process is in place for requirements change management?
4. How are estimations captured?
5. What process is documented for defect management?
6. How are documents created and managed?
7. Describe how code reviewed are completed
8. What role does unit testing have in the development process?
9. How often is code checked-in, is there a guideline for checking in code?
10. What automated build procedures are used if any?
11. How are individual components tested?
12. Describe how system integration is completed?
13. What automated testing is in place?

Appendix II

FPGA for Telco Case Study Three Interview Questions:

- Describe your role?
- Who do you interact with most with in your role?
- Describe what project success looks like to you?
- What is your definition of software quality?
- In relation to the FPGA Telco what was the single most important factor in the program?
- What is the one aspect of the FPGA Telco program you would change?


http://www.sagepub.com/chambliss4e/study/chapter/encyc_pdfs/4.2_Triangulation.pdf.


Committee, S.E.S., 1998. IEEE Recommended Practice for Software Requirements Specifications 830-1998,

Creswell, J.W., 2013. Qualitative Inquiry and Research Design,

Croll, P.R., Drive, C. & George, K., 2008. QUALITY ATTRIBUTES - ARCHITECTING SYSTEMS TO MEET CUSTOMER EXPECTATIONS CsC.


Date, P. & It, T., 2014. Increased Complexity in Embedded Software


LinenShapeSpace, 2015. 3 Excellent Examples of Corporate Social Responsibility.


Seo, M.-G., 2005. Understanding the Human Side of Merger and Acquisition:


Projects: Developing Reliable Scales to Predict Multiple Perspectives by Multiple. (October).


