A Quality Framework for Software Development (QFSD)

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Additional Information:

- A Doctoral Thesis. Submitted in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy at Loughborough University.

Metadata Record: https://dspace.lboro.ac.uk/2134/36013

Publisher: © William Peter O’Neill

Rights: This work is made available according to the conditions of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International (CC BY-NC-ND 4.0) licence. Full details of this licence are available at: https://creativecommons.org/licenses/by-nc-nd/4.0/

Please cite the published version.
Please note that fines are charged on ALL overdue items.
A Quality Framework for Software Development (QFSD)

By
William Peter O’Neill
A Doctorial Thesis
Submitted in partial fulfillment of the requirements for the award of
Doctor of Philosophy Loughborough University
April 2010
© by William Peter O’Neill (April 2010)
Thesis Abstract

Introduction

This research delivers a new complete and prescriptive software development framework, known as The Quality Framework For Software Development (QFSD) for immediate use by software development practitioners. Whilst there are a number of existing methodologies available, and many software development standards they fail to address the complete development lifecycle. A review of current literature supports this assertion.

Aims & Objectives

The overall aim of the research is to create a new software development framework, applying it to a substantial number of real world software projects in two different industrial software development environments and thereby demonstrating its effectiveness.

Methods

Based on a review of the available research approaches and strategies, the researcher selected 'pragmatism' as the most suitable for this research. This selection was driven by two contributory factors. The first was that in order to conduct the research the researcher would have active participation in the majority of the research activities. The second was that the deliverables from the research should be immediately useable for the benefit of software practitioners and hence not be regarded as a theoretical framework. The approach was further refined by adopting Action Research and Case Study strategies. The research was divided into stages each of which was executed within separate companies. The companies were very different in terms of their business areas, culture and views on quality and specifically quality of software deliverables.

Results

The research findings provided a strong indication that a holistic software development framework does provide an improvement in software project deliverables quality and repeatability in terms of schedules and quality. In the case of Fisher-Rosemount it enabled them to attain ISO 9000/Ticket accreditation. In addition, by providing all processes and tools in a single web based environment the adoption by software developers, project managers and senior management was very high.
Table of Contents

TABLE OF FIGURES ...................................................................................................................... 6
TABLE OF TABLES .......................................................................................................................... 7

1 CHAPTER 1 - INTRODUCTION .................................................................................................. 9
   1.1 INTRODUCTION .................................................................................................................... 9
   1.2 RESEARCH GOAL ................................................................................................................. 9
   1.3 RESEARCH RATIONALE ....................................................................................................... 10
   1.4 AIM ...................................................................................................................................... 13
   1.5 RESEARCH OBJECTIVES ................................................................................................... 13
   1.6 THESIS STRUCTURE .......................................................................................................... 14

2 CHAPTER 2 - REVIEW OF LITERATURE AND EXISTING SOFTWARE PROJECT MANAGEMENT METHODOLOGIES ................................................................. 17
   2.1 INTRODUCTION .................................................................................................................... 17
       2.1.1 The need to follow a methodology ................................................................................. 17
       2.1.2 Scenario one .................................................................................................................. 19
       2.1.3 Scenario two .................................................................................................................. 19
       2.1.4 Scenario three ............................................................................................................... 21
   2.2 THE NEED FOR A NEW FRAMEWORK .............................................................................. 24
       2.2.1 Does existing literature indicate that there are issues with current methodologies? ...... 25
       2.2.2 Why is an integrated framework covering all aspects of development a good thing? ...... 26
       2.2.3 Why is there a need for it to be flexible? ........................................................................ 27
       2.2.4 Why is some tailoring for different quality targets necessary? ...................................... 27
       2.2.5 Why is reuse a good thing? ............................................................................................ 29
       2.2.6 Why is a supporting tool set necessary? ......................................................................... 31
       2.2.7 Why is communication a key factor? .............................................................................. 32
       2.2.8 What key processes need to be supported by the tool-set? ............................................ 33
   2.3 EXAMINATION OF EXISTING METHODOLOGIES ......................................................... 36
       2.3.1 Agile Methodologies ...................................................................................................... 37
       2.3.2 Process Assessment Frameworks ................................................................................... 39
       2.3.3 DOD-STD, MIL-STD & RUP Processes ...................................................................... 45
   2.4 LITERATURE REVIEW CONCLUSION ............................................................................ 48

3 CHAPTER 3 - RESEARCH METHODOLOGY ............................................................................. 52
   3.1 INTRODUCTION .................................................................................................................... 52
   3.2 RESEARCH PHILOSOPHY ..................................................................................................... 52
       3.2.1 The Positivist Approach ............................................................................................... 53
       3.2.2 The Anti-Positivist Approach (Interpretivism) ............................................................... 53
       3.2.3 The Pragmatic Approach (Pragmatism) ........................................................................ 54
   3.3 RESEARCH APPROACH ....................................................................................................... 56
       3.3.1 Research environment .................................................................................................... 56
       3.3.2 Research approaches considered .................................................................................. 57
       3.3.3 Research strategies considered .................................................................................... 58
   3.4 CONDUCTING THE RESEARCH ......................................................................................... 74
       3.4.1 Definition of phases ....................................................................................................... 75
   3.5 RESEARCH ROADMAP ....................................................................................................... 77
   3.6 CONCLUSIONS .................................................................................................................... 78

4 CHAPTER 4 - ARE CURRENT QUALITY STANDARDS SUFFICIENT FOR QUALITY SOFTWARE? ......................................................................................................................... 82
   4.1 INTRODUCTION .................................................................................................................... 83
7 CHAPTER 7 - CASE STUDY THREE: VALIDATION OF THE CORE QFSD MODEL AT FISHER-ROSEMOUNT

7.1 INTRODUCTION ......................................................................................................................... 186
7.2 CONCLUSION ............................................................................................................................. 189

8 CHAPTER 8 - PROCESS PERFORMANCE AND CAPABILITY TECHNIQUE ........................................................................................................... 191

8.1 INTRODUCTION .......................................................................................................................... 192
8.2 PROCESS PERFORMANCE MEASUREMENT METHODOLOGY ......................................................... 193
8.3 PROCESS PERFORMANCE ANALYSIS METHODS ......................................................................... 196
8.4 ACTUAL MILESTONE CONTROL DIAMOND MEASUREMENT RESULTS ............................................. 200
8.5 PROCESS CAPABILITY MEASUREMENT METHODOLOGY ........................................................... 205
8.6 EXAMPLE OF THE PROCESS CAPABILITY ANALYSIS .................................................................... 208
8.7 CONCLUSIONS ............................................................................................................................. 211

9 CHAPTER 9 - ADAPTABILITY & COVERAGE OF THE QFSD FRAMEWORK ......................................... 213

9.1 INTRODUCTION .......................................................................................................................... 214
9.2 PHASE ONE: INTRODUCTION OF EMERGENCY PROCESSES ..................................................... 215
9.3 PHASE TWO: ESTABLISH QUALITY FRAMEWORK REQUIREMENTS TOGETHER WITH THE NEW IT ORGANISATION AND INTRODUCE THE BASIC QFSD FRAMEWORK ........................................................................................................... 218
9.3.1 Establish quality framework requirements ................................................................................. 218
9.3.2 New IT organisation .................................................................................................................. 220
9.3.3 Introduction of the basic QFSD framework ................................................................................. 222
9.4 CONCLUSIONS ............................................................................................................................. 237

10 CHAPTER 10 - SCALABILITY OF THE QFSD FRAMEWORK ................................................................ 241

10.1 INTRODUCTION ......................................................................................................................... 242
10.2 INTRODUCTION OF PRINCE2 ................................................................................................... 243
10.2.1 Adoption of the PRINCE2 templates ....................................................................................... 244
10.2.2 Adoption of PRINCE2 management processes ....................................................................... 246
10.2.3 Pilot of PRINCE2 integration with QFSD ................................................................................. 247
10.2.4 Conclusions from the PRINCE2 pilot ...................................................................................... 252
10.3 INTRODUCTION OF RUP ......................................................................................................... 253
10.3.1 Review and integration of RUP ............................................................................................... 254
10.3.2 Pilot of RUP integration with QFSD ....................................................................................... 268
10.3.3 Conclusions from the RUP pilot ............................................................................................. 272
10.4 CONCLUSIONS ............................................................................................................................. 273

11 CHAPTER II - CONCLUSIONS AND FURTHER RESEARCH ............................................................ 279

11.1 INTRODUCTION .......................................................................................................................... 279
11.2 ACHIEVING THE OBJECTIVES ................................................................................................. 279
11.2.1 Objective One .......................................................................................................................... 280
11.2.2 Objective Two .......................................................................................................................... 281
11.2.3 Objective Three ....................................................................................................................... 284
11.2.4 Objective Four .......................................................................................................................... 287
11.2.5 Objective Five .......................................................................................................................... 289
11.3 FUTURE RESEARCH AREAS ..................................................................................................... 289
11.3.1 Application of QFSD to European programmes involving business change ......................... 290
11.3.2 Integration of Cassandra into the framework ........................................................................... 290
11.3.3 Integration with an SOA service repository ............................................................................. 293
11.3.4 Configuration Management and Version Control ..................................................................... 300
11.4 OVERALL CONCLUSION ............................................................................................................. 301
Table of Figures

Figure 2.1: The Position of Agile Processes on the ProcessMap ........................................ 38
Figure 2.2: The Assessment Frameworks on the ProcessMap ........................................... 44
Figure 2.3: Military Standards and RUP on the ProcessMap ............................................... 48
Figure 3.1: Checkland’s soft systems methodology inquiry process ...................................... 69
Figure 3.2: Thesis roadmap .................................................................................................... 77
Figure 3.3: A basic learning cycle ........................................................................................ 79
Figure 4.1: Staff Profile for Case Study ................................................................. 102
Figure 5.1: QFSD Framework Model .................................................................. 112
Figure 5.2: Five fundamental steps in establishing a quality framework ............. 116
Figure 6.1: QFSD framework validation and progress chart – tools & techniques ... 139
Figure 6.2: Celesio QFSD repository architecture ............................................. 149
Figure 6.3: Example QFD matrix structure ....................................................... 169
Figure 6.4: Overall testing hours’ profile ........................................................... 178
Figure 6.5: Total numbers and classifications of defects discovered .................... 179
Figure 6.6: Total defects discovered rate graph ................................................ 180
Figure 6.7: Critical defects discovered rate graph .............................................. 181
Figure 7.1: Framework validation and progress chart – second case study ........... 186
Figure 7.2: Structure of the core QFSD framework .......................................... 187
Figure 8.1: Framework validation progress chart - process performance and capability 191
Figure 8.2: Open verses Closed Loop Process Control .................................... 194
Figure 8.3: Control Diagram Example ............................................................... 197
Figure 8.4: Control Chart for UI Cluster Estimates ........................................... 202
Figure 8.5: Number of Sub Task Estimate Accuracy Values at Different Standard Deviations for the UI Cluster .................................................. 204
Figure 8.6: Bar Chart completion levels for project planning process of each work product 209
Figure 9.1: Lifecycle model ............................................................................. 229
Figure 10.1: Framework validation and progress chart – Commercial environment 234
Figure 10.2: QFSD, PRINCE2 & RUP Mapping ............................................... 255
Figure 10.3: Alignment of RUP Phases with QFSD Phases ............................... 258
Figure 11.1 Cassandra dependency diagram .................................................... 292
Figure 11.2 Service Lifecycle ......................................................................... 296
Figure 11.3 Role of the Service Repository in the development process ............ 298
Figure 11.4 Run-time context showing UDDI Registry .................................... 299
Figure B1: Celesio Option 2 ............................................................................ 333
Figure B2: Celesio Option 3 ............................................................................ 334
Figure M1: The project planning process ......................................................... 413
Figure M2: Third Case Study staff profile ....................................................... 415

Table of Tables

Table 2.1 Project Success Factors ................................................................. 34
Table 2.2 Project Challenged Factors ............................................................. 34
Table 2.3 Project Impaired Factors ................................................................. 35
Table 4.1 Summary of Project Management Questionnaire ............................ 89
Table 5.1 QFSD initial framework validation project history .................................................. 115
Table 5.2 Core QFSD Processes Model ............................................................................... 120
Table 5.3 Organisational process descriptions ..................................................................... 124
Table 5.4 Management process descriptions ....................................................................... 127
Table 5.5 Design principle framework mapping ................................................................. 135
Table 6.1 Cluster Goals ......................................................................................................... 171
Table 8.1 Milestone / Schedule dates .................................................................................. 201
Table 8.2 Process Performance Analysis Summary ............................................................. 204
Table 8.3 Planning work product analysis report ..................................................................... 208
Table 8.4 Capability analysis results for the Control Desktop p2.0 project ............................. 210
Table 9.1 Phase one project delivery analysis ....................................................................... 217
Table 9.2 Overall lifecycle phase model ............................................................................... 227
Table 9.3 Phase two project delivery analysis for selected projects ..................................... 232
Table 9.4 Phase two cluster process performance analysis - Time Management System .... 233
Table 9.5 Phase two cluster process performance analysis - Pharmacy Point Europe ........ 234
Table 9.6 Phase two project process capability analysis - Time recording system ............... 235
Table 9.7 Phase two project progress capability analysis - Pharmacy Point Europe ............ 236
Table 10.1 PRINCE2 artefact template mapping ................................................................. 245
Table 10.2 PRINCE2 process mapping .................................................................................. 246
Table 10.3 Phase three step one project delivery analysis ..................................................... 248
Table 10.4 Phase three step one cluster process performance analysis – Invoice Matching ... 248
Table 10.5 Phase three step one cluster process performance analysis – Weldrick’s .......... 249
Table 10.6 Phase two step one project process capability analysis – Invoice Matching ...... 250
Table 10.7 Phase two step one project process capability analysis – Weldrick’s ................. 251
Table 10.8 QFSD Phase to RUP Phase Mapping ................................................................. 259
Table 10.9 Combined RUP and QFSD Roles ....................................................................... 262
Table 10.10 Combined RUP and QFSD / PRINCE2 products and artefacts ......................... 264
Table 10.11 Phase three step two project delivery analyses ............................................... 268
Table 10.12 Phase three step two cluster process performance analysis Customer Profitability 269
Table 10.13 Phase three step two cluster process performance analysis – Win FSM ............ 269
Table 10.14 Phase two step two project process capability analysis Customer Profitability 270
Table 10.15 Phase two step one project process capability analysis – Win FSM ............... 271
Table A1 - A chronological history of the first case study ..................................................... 322
Table C1 - Core QFSD processes ......................................................................................... 337
Table F1 - Lifecycle phases & artefact mapping ..................................................................... 387
Table G1 - Case Study Summary .......................................................................................... 391
Table H1 - Mapping of QFSD processes against ISO 9001 and CMM ................................. 396
Table I1 - QFSD process contribution relevance questionnaire ........................................... 397
Table J1 - QFSD user survey questionnaire ......................................................................... 398
Table M1 - Estimate refinement technique ........................................................................... 418
1 Chapter 1 - Introduction

1.1 Introduction

This chapter introduces the research goal of the thesis leading on to an outline and brief examination of the research rationale. The aim of the thesis, the creation of a new framework for software development, is then stated. Objectives are proposed that will lead to achieving this aim. The final section provides an outline of the thesis itself.

1.2 Research Goal

Experience over the past 20 years in the software industry shows that obtaining high quality software in a predictable timescale and at an acceptable cost, has proved illusive for many companies (Jones, 1995).

In order to address this critical issue a number of methodologies have been applied over the years. For example; ISO9000/TickiT (The TickiT Guide, 2001, Issue 5.0), CMM (Caputo, 1998), Crosby's methods (Crosby, 1986), Prince 2 (Bradley, Franklin, 2003) Rational Unified Process (Kruchten, 1996) etc. All are effective at setting the objectives and broad principles for achieving successful projects; however they all suffer from a common problem. Fundamentally, they describe what the objectives for a successful project are, but not how to achieve those objectives in a practical sense. They do not provide a sufficient guide for creating an environment for achieving successful, quality software.

Based on extensive experience in software development, management and delivery, it is the researcher's belief that a more comprehensive software development framework can be developed, which could help to improve the probability of delivering successful software projects.

The overall aim of this thesis is to create a new framework for software development, based on an iterative approach enhancing and verifying the framework by applying it in a controlled manner to a significant number of commercial software projects in different software development departments.
the application and refinement of the framework over multiple projects and in multiple development departments the effectiveness of the framework in a diverse software industry context should be observable. For the purposes of this thesis, a software project development framework is a collection of processes, methods and tools for use for software project development. For any given software project it would not be necessary to use all the processes, methods and tools in the framework as they would not all be appropriate in any specific case. Associated with the framework, therefore, is a methodology to select and use the appropriate parts of the framework on a given project. In this research, the developed framework has been named the “Quality Framework for Software Development”, which is usually shortened to “QFSD”. The framework is referred to as the “QFSD framework”, or simply “QFSD” and the associated methodology is referred to as the “QFSD methodology”.

1.3 Research Rationale

The immediate question that arises is ‘why another methodology’? If we look at the many published papers for software project deliveries Alter, Ginzberg (1978), Barki et al, (1993), Boehm (1991), Charette, (1989) & McFarlan (1981), the failure-to-complete rate is close to 13% and deliveries that fail to meet user expectations or overrun budgets are greater than 60%, which means close to 25% of projects actually come in on time and to budget (Stang, 2004). The result is that over half the software projects executed either do not fulfil user expectations or overrun. In either case this has a substantial impact on overall costs, either by direct overspend against budget or rework to modify the software in order to meet the original user expectations.

The cost of this continued failure to deliver software projects on time, to budget and at the right level of quality, where quality includes user expectations, is a source of reduced company profits and results in a lowering in confidence of software solutions. This lowering of confidence by business users drives down future investment in IT, which would have benefited companies' performance. It
contributes to the continued issues of legacy system support and continued cost cutting of IT budgets until they hit a level were low quality software is inevitable.

Do software failures only occur in companies that have not adopted one of the current methodologies? Software projects do fail in companies that have adopted development methodologies, but the reasons are interesting. Gartner published a paper (Stang, 2004) that indicates that whilst companies may have adopted development methodologies they still suffer from poor overall management of their software development processes. For example, even with strong development processes a company must also manage the following:

- Multiple IT projects and associated budgets
- Selecting which projects to fund
- Maintaining a skills repository and usage repository
- Demonstrating sufficient return on investment

Gartner carried out an analysis across a number of commercial software companies and reviewed the results from 378 projects.

These projects were grouped in to investment bands as follows:

- Greater than $1M
- Between $0.5M to $1M
- Between $100 to $0.5M

With the result that:

- In the greater than $1M group, all projects came in roughly on time and budget.
- In the $0.5M to $1M group only 78% of projects came in roughly on time and budget.
- In the $100 to $0.5M group only 28% of project came in roughly on time and budget.
It is also useful to observe that out of the 51 projects that were abandoned, 47 of them were in the $100 to $0.5M group.

The conclusions were that larger projects receive the best skilled resources, maximum quality control, and senior management exposure. In contrast, the smaller projects were often assigned resources with lower experience level or were categorised as lower priority for those highly skilled resources working on the larger projects. The smaller projects were also found to have a much-reduced level of quality processes applied to them.

Clearly, managing a number of smaller projects with less skilled resources is a major challenge, but one that most IT groups have to meet. However, companies that adopt quality standards such as TickIT are given guidelines on elements such as requirements management and review procedures, but no real structure in which to introduce the improvements and certainly no idea on how to actually implement them (Mingay, 2003).

The results tend to be an increase in overhead costs in order to generate the documentation required by the standard, but no real improvement in time to market or user expectations. A case study that illustrates this is included in the next chapter.

This highlights the need for a practical software development framework be introduced, which provides the following:

- Practical instructions on how and when to use components of the framework
- Scalability in terms of project size or programme of projects and complexity
- A method for the determination of which projects to actually run
- Flexibility to allow improvements to be introduced progressively and at a practical level
- A clear alignment with the objectives of the business users
- A practical tool in which the artefacts of the framework can be located and managed
• The ability to tailor and add to the framework based on the specific company’s environment

1.4 Aim
The aim is to create a new software development framework, applying it to a substantial number of real world software projects in two different industrial software development environments and thereby demonstrate its effectiveness.

1.5 Research Objectives

The main objectives of the research are as follows:

1. To create a practical and somewhat prescriptive software development framework and associated methodology for applying the framework, based on repeated application and adjustment of the framework over a substantial number of software development projects and verified against published evidence.

2. To validate the framework by using it on a number of real projects, initially in a technical development company environment.

3. To test the scalability and applicability of the framework by subjecting it to a large commercial software environment and, subsequently, to ever increasing project types, sizes and complexities.

4. To refine the software project development framework in the light of experiences of tests in objective 3. Objectives 3 and 4 will be continued in an iterative, evolutionary manner.

5. To develop a proto-type repository tool-set to support the framework, methodology and project artefacts. This will be carried out in parallel to objectives 1 to 4.

Section 3.4 "Conducting the research" outlines the research approach adopted in order to achieve the five objectives.
1.6 Thesis Structure

This thesis is composed of eleven chapters.

Chapter Two examines published literature concerning existing software project and development management methodologies, concluding that a more practical and prescriptive software product development framework would help ensure that software deliveries were more predictable and of higher quality.

Chapter Three evaluates the applicable research methodologies and concludes that research philosophy of 'pragmatism', together with an idiographic approach and using a combination of strategies: Contextualism, Grounded Theory, Action Research and Case Studies are the most appropriate. The Chapter also provides a roadmap showing the research activities in relation to achieving the major framework design principles.

Chapter Four considers the extent to which today's current methodologies contribute to software quality, which is further illustrated with a detailed case study. Based on the literature search in the previous chapter and the detailed case study a set of design principles are defined on which the proposed new framework could be developed.

Chapter Five provides an overview of the QFSD framework core design, taking the design principles in to account. The QFSD Core Framework is then verified again a number of real projects within the researcher’s previous company Fisher-Rosemount.

Chapter Six describes the creation and application of the main supporting tools and techniques that have been used in the verification and application of the QFSD framework.

Chapter Seven describes the application of the QFSD Core Framework to a large new software product development at Fisher-Rosemount. It shows the effectiveness of the framework and enables the practitioner to better understand how the framework is used in practice.
Chapter Eight describes the first two major extensions to the framework, which enable the performance and capability of each framework process to be measured, verified and continuously improved.

Chapter Nine outlines the challenge faced when introducing the framework into a larger commercial software development organisation that had no previous experience of quality standards or any professional software engineering processes.

Chapter Ten addresses the integration of industry standard methodologies into the framework and provides a generalised approach for the integration of other methodologies.

Chapter Eleven revisits the aims and objectives and shows how the original objectives were addressed. The chapter also includes details on areas of work that were not fully successful, how the approach was changed in some cases and lessons learned factored into the final QFSD Framework. Ideas for follow-up areas of research are also included.

Appendices included are as follows:

- Appendix A: “Case study Fisher-Rosemount”. This contains detailed results from the first validation of the core QFSD Framework.
- Appendix B: “Example of failure in new product development using waterfall methodology”. This is a case study taken from Lloyds Pharmacy showing the impact of using a waterfall methodology inappropriately.
- Appendix C: “Core QFSD process model”. This contains the definition of all core QFSD processes.
- Appendix D: “Project management questionnaire results”. This is the questionnaire taken from White and Fortune (2002).
- Appendix E: “Project management plan template”. This is the extensive project management plan template taken from the QFSD repository.
• Appendix F: "Lifecycle phase mapping". This gives an example of mapping a lifecycle model to the QFSD processes and selecting the required deliverables from each phase.

• Appendix G: "Case Study Survey and Limitations". Provides a summary of the approach and techniques used in all case studies, together with the limitations and lessons learned.

• Appendix H: "QFSD to ISO 9001 & CMM mapping". Provides the analysis carried out in order to establish that the framework provided the same coverage of the software lifecycle as a key subset of current software quality standards.

• Appendix I: "QFSD process contributor relevance questionnaire". Provides the survey template used to determine the major stakeholders associated with a typical software development and the process reporting areas required.

• Appendix J: "QFSD user survey questionnaire". Provides the survey template used to monitor the actual use and success / issues in both the development and user communities.

• Appendix K: "QFD matrix overview". Provides the hierarch of matrix analysis used in the application of QFD as a requirements gathering and prioritisation method.

• Appendix L: "Beneficiaries questionnaire". Provides the survey template used as part of the QFD requirements gathering method.

• Appendix M: "Case Study Three". Details of Case Study Three have been included in an appendix as this is aimed at the software practitioner's detailed application of the framework and therefore does not appear in the main Chapters of the thesis.
2 Chapter 2 - Review of Literature and Existing Software Project Management Methodologies

This chapter examines the literature concerning published software project management and development methodologies. Existing methodologies are examined and the contribution of each is compared. The views of independent researchers on these methodologies are also evaluated and compared with that gained by experience at the companies employing the researcher of this thesis. The conclusion examines the aims of the research and identifies the contribution the research will make for software developers and project managers. As this research is based on pragmatism, it seeks to deliver a practical and immediately useable framework for IT practitioners.

This chapter reinforces the objective of why a new practical and somewhat prescriptive software project development framework, based on proven and real world experience is required.

2.1 Introduction

Before examining the existing methodologies it is worth considering two questions:

- Why is a methodology required in the development of software products?
- Why do such methodologies benefit from being part of a development framework supported by an appropriate level of tools?

2.1.1 The need to follow a methodology

Is there a need to follow a methodology? At first glance this would seem to be a naïve question. However, if we place ourselves outside the software industry and take the view of a business user, we see that the press received by the software industry is not glowing and we can forgive them wondering if we in the IT industry follow any methodology at all. For example, in recent years we have seen failures in the National Programme for IT in the NHS (National Audit Office),
London Stock Exchange (Charette, 2005), US Internal Revenue Service Modernisation (Charette, 2005), Washington State Drivers and Vehicles Registration System (Charette, 2005) etc. The question therefore would be more realistically formulated as “does the software industry use any methodology at all?”. The easy answer is that many methodologies are used, but the extent to which they cover all related activities and the quality to which they achieve this, vary between methodologies.

Methodologies need also to be applicable to the type of software development being undertaken. For example, the waterfall methodology works acceptably well in a project that is required to make a series of updates to an existing software product. An existing product would be stable in terms of technical design and would normally have functional requirements that do not require major redesigning of the applications structure. However, in developing a new software product from scratch, the waterfall model has proved to be inadequate in the researcher’s experience. Laplante (2004) supports this idea and further suggests that the waterfall approach is not applicable in developments that require close and constant communication with the customer. Taking this a step further, even the application of iterative methodologies can fail to deliver when applied incorrectly, according to Kruchten (2007).

In order to illustrate the above points consider the planning, design and construction of a building as detailed in the following three scenarios.

This is a well used analogy, for example Winograd et al, (1996) used the analogy to emphasis that architecture design in software production is a major success factor. However, the researcher has updated the analogy to take account of a major new user requirement namely software service (component in old object-oriented terminology) reuse. Each part of the analogy will be reinforced by recent projects within the researcher’s company Celesio.
2.1.2 Scenario one

Scenario one considers the construction of a typical modern detached town house, typically built as part of an estate of such houses. This type of construction is based on a simple construction plan, standard list of materials, known set of skills, required effort and typical build time. Each constructed instance is almost identical to the next.

The objective is to be able to build as many of these town houses as quickly and cheaply as possible. As such, there are few variables in the planning, design and construction. Those variables that exist are limited to carpets, kitchen fittings, door types and sometimes garden layout. They are not subjected to major requirements changes during the build, such as moving the bathroom downstairs, adding an en-suite to a bedroom or building a double, rather than a single garage. Therefore, a simple waterfall methodology would suit this type of construction as the build in each case is very similar and hence the team understands what is expected, and no redesign, no requirement changes and hence very little project management needs to be applied.

An iterative methodology could equally be applied, but given the simple and well understood repeated activities, this would really be an iterative method applied following a waterfall approach, i.e. each phase in the iterative approach would be applied only once and hence it would effectively be the same as a waterfall approach, but using different terminology.

2.1.3 Scenario two

Scenario two considers a bespoke detached town house, being built for a specific customer with a generous funding, but a high quality level expectation. Whilst an initial construction plan, list of materials, skill set and estimates of effort and build time exist, it is clear that the requirements are evolving and indeed some of the requirements will be changed as the build progresses. However, the
client is willing to accept increased costs, but no relaxation on the completion date.

In this case it is likely that major changes will be requested during the construction phase and these need to be built in to the design and order of construction. For example, the requirements, design and construction need to have a level of flexibility built in to them. Construction needs to be carried out in a way that minimised the costs of change e.g. all piping needs to be above the floor level and not under concrete. All walls need to be in place for each floor, using simple stud walls and reviewed with the customer before any permanent fixing is completed.

Essentially the requirements, design and construction are carried out as a set of iterations, each being communicated and reviewed by the customer. Iterations can be executed in parallel in order to adsorb additional requirements, but the dependencies between iterations need to be closely managed. This type of construction is suited to an iterative methodology as the requirements are evolving, change is part of the design brief and hence more sophisticated project management is required.

The following is an example of what can happen if a waterfall method is applied to a new bespoke (from scratch) software product development and the catastrophic result can be seen in Appendix B.

The example in Appendix B is taken from a new product development project carried out between March 2002 and June 2003 by Lloyds Pharmacy in the UK. The new product was required to replace an existing store based C language & DOS operating system application, which exchanged information between the stores and head office by using a simple dial-up polled mechanism that ran each evening.

The new product was a thin browser based in-store client with the servers and hence majority of processing designed to be carried out centrally. However, the first attempt at the development of this product by an external partner using the waterfall method was a very expensive and time-consuming failure. The supplier
claimed that they were using the Rational Unified Process RUP, but in fact they were simply using the phases from RUP, but applying them in a linear fashion, which, in effect, is the waterfall approach.

The project was restarted taking into account the recommendations from the IBM report as detailed in Appendix B. The researcher led the project review and had responsibility for approving the restart project approach and planning. The researcher set the following four objectives for the project, which were based on the corporate standard development methodology:

- That the project use the PRINCE 2 project management methodology
- That a new software subcontractor be selected following a rigorous software quality process audit
- That a more achievable and robust architecture be considered that would be more appropriate to a retail store environment
- That the RUP software development methodology be used in order to deliver the new software product using a standard iterative development method, which would be familiar to both Lloyds Pharmacy and the new software supplier

Following the introduction of the above changes to project approach, the project managed to keep to its new project plan and met its delivery dates. A major observation is that due to the application of the above the project became much more predictable with less pressure on the actual project team.

A further observation is that software development frameworks must be adaptable to the type of development and be capable of supporting industry standard methodologies. This supports the framework design requirement 5 described in Chapter 4.

2.1.4 Scenario three
Scenario three considers the case where the blueprint planning, design and construction approach is being created for the standard detached mass market house as described in scenario one. Basically, this is the creation of a bespoke house plan, design and construction, so has an extensive set of pilot builds, which will then be built multiple times by mass market housing construction companies. In other words the planning, design and construction must be delivered as a set of reusable services (components). However, unlike the house in the first scenario, this house design is upmarket and hence has a number of major configuration options such as ceiling height, single or double garages, conservatory types, roof design etc.

In this case, application of an iterative development methodology is not sufficient. The design must be capable of being reused and hence the effort in the design and pilot phases is more extensive and needs to address the design and construction of each major space in the house such that the dependencies, being the spaces, are as decoupled as possible, thus allowing the maximum configuration flexibility. The same approach needs to be applied to 'technical services' such as mains water, gas and electricity in order that the spaces can be serviced easily no matter what space configuration is defined by the end customer (this equates to a service bus concept).

The modern design philosophy for achieving this level of reusability is part of Service Oriented Architecture (SOA). Whilst SOA is not a software development methodology in itself, it does rely heavily on a suitable development methodology to enable it to achieve the objective of software services reuse, both within a given application and between different applications. This leads to the concept of SOA governance, which is simply a high quality software development and release management environment required to achieve full benefits from an SOA approach. Whilst it is not the researcher's intention to discuss SOA in detail, it is worth noting that failure to apply an adequate level of SOA governance will result in a failed SOA initiative and make reuse of developed services unlikely. Manes (2005) goes further, stating "SOA governance is critical to the success of an SOA initiative". Burton Group predicts that businesses will waste millions of dollars
over the next three to five years in failed efforts to adopt SOA because they won't institute the necessary governance processes to ensure that things are developed with some "sense of purpose and discipline". Again a lack of a suitable methodology may cause the software industry to relive the past pains experienced when adopting new technology.

Whilst Manes (2005) supports the idea that governance is one of the keys to a successful SOA, the researcher has witnessed a real project that has already added to the assertion. Celesio initiated an SOA project to deliver a new Order Taking application for the placing of orders from pharmacies to Celesio warehouses, using many different ordering protocols (essentially an internet ordering portal). The project also had the objective of establishing an SOA development and operating environment, by establishing the necessary skills and tools across the three major business units in the group (United Kingdom, France & Germany).

The project suffered from a number of management and execution issues, but the most relevant for this example is that it failed to deliver the following key items:

- Did not use a UDDI (Universal Description, Discovery and Integration) registry, which meant that there was no single point of control for publication, quality, integrity or discovery of services

- Did not use a development service repository, which meant the provision of services, changes, versioning and discovery was not possible

- Did not document or publish the developed services in a way that could be used by developers in order to achieve reuse

- Did not apply any form of reuse approach in the design of the services

The project failed to use the Celesio standard project management approach and this caused its own issues. The project did use an iterative approach, but failed to extend the methodology to incorporate the development policies required to make a successful SOA project.
Strong methodologies should be used within an integrated, scalable and adaptable framework; this is supported by Gibson et. al (2006), Baker et al (2007), Butler (1995) and McGary (1998). Therefore one methodology does not fit all cases, but a common framework in which specialist methodologies and best practices can be added is the preferred approach. This supports framework design principles 4, 5 and 10 described in Chapter 5.

2.2 The need for a new framework

The research described in this thesis is based on the assertion that a common framework that provides the fundamental best practices for software production, coupled with the capability to integrate with existing industry best practice methodologies and tool-sets, promotes a continuously improving and productive software development environment.

However, is this view generally supported?

In order to test this we need to consider the following questions:

1. Does existing literature indicate that there are issues with current methodologies?
2. Why is an integrated framework covering all aspects of development a good thing?
3. Why is there a need for it to be flexible?
4. Why is some tailoring for different quality targets necessary?
5. Why is reuse a good thing?
6. Why is a supporting tool set necessary?
7. Why is communication a key factor?
8. What key processes need to be supported by the tool set?
9. Does existing literature indicate that there are issues with current methodologies?
2.2.1 Does existing literature indicate that there are issues with current methodologies?

In a paper by Fitzgerald B (1996) a critical analysis of formal software development methodologies was carried out. The research indicated that practitioners are moving slowly towards adoption of formal methodologies. However, it also suggests that the trend in literature to highlight the failure rate of software projects may be a significant contributory factor in the improving adoption rates.

The research further highlights the extent to which literature supports the need for adoption of improved methodologies by highlighting the following examples:

- The major forces working for diversity appears to be ignorance, lassitude, deficiency...Employers have been very relaxed about setting and enforcing local standards for their employees to follow. (Chapin, 1981)
- The problems faced in developing large software include...enforcing a methodology on the developers. (Ramamoorthy et al., 1986)
- The first effect of teaching a methodology—rather than disseminating knowledge—is that of enhancing the capacities of the already capable, thus magnifying the difference in intelligence. (Dijkstra, 1972)
- Losers consist of unnamed, unspecified, up to the individual, or 'written-but not formalised' types of methodologies. (Zolnowski & Ting, 1982)
- The use of a formalised SDM is perceived as positive and well advised. (Jenkins et al., 1984)
- Software development in professional communities often is completely ad hoc, or at best supported by structured methods...and JSD. (Plat et al., 1991)
- One startling and somewhat disturbing observation is that many (systems development) methods are used very little (Palvia & Nosek, 1993)
These are a small sample that illustrates the general view in the literature concerning the short comings of current methodologies.

2.2.2 Why is an integrated framework covering all aspects of development a good thing?

Jones (1995) indicated that adoption of a fully integrated software development environment (I-Case, integrated computer aided software development environment) resulted, based on his extensive evaluation of hundreds of software projects, in a 50% improvement in quality levels and a 35% improvement in software productivity. This supports framework design principle 6 described in Chapter 5. However, commercial I-Case tool suits are extremely expensive ranging from £5K to £10K per seat. For smaller IT organisations this is a cost issue, which is compounded by the extensive training that is necessary in order to use the environment competently and start to achieve improvements and hence show a return on investment.

Closer examination of these tools reveals that I-Case tools and ISEE tools do not cover the entire software design lifecycle and most notably lack the necessary integration between project management and development processes. Therefore a fully integrated framework is still not available, which covers the whole software lifecycle and is affordable for the majority of IT organisations. However, it should be understood that software development tools are not a 'silver bullet' (Brooks, 1995) that will ensure success; rather it's the software development staff's understanding and correct application of the processes that is the key. These observations are supported by Jones (1995), who indicates that I-Case tools generally lack support for quality control, document management and security and that tools do not, in themselves improve quality, but rather the training and application by development staff. However, tools are a key factor in promoting the use of correct processes. This supports framework design principle 6 described in Chapter 5.
2.2.3 Why is there a need for it to be flexible?

Flexibility in this context refers to the ease with which a framework can be adapted to include a new methodology. For example, whilst the framework may support a standard project management methodology, best practices and have a core information repository, it should also have the capability to easily integrate a new requirements modelling method, such as UML diagramming and communicate with a commercial UML modelling tool.

- The philosophy (which supports framework design principle 5 described in Chapter 5) should be to establish the overall framework in which an IT organisation wishes to manage its software development and reflect this as a group of easily accessible polices in a core data repository. However, once established, the framework should be populated with industry methodology tools if possible, such as the Rational Unified Process or PRINCE 2 policies, if that becomes better suited to the IT department's needs (Zarrella, 1990), (Thomas, Nejmeh, 1992). This supports framework design principle 5 in Chapter 5.

2.2.4 Why is some tailoring for different quality targets necessary?

Baker et al (2007) support the assumption that one of the major keys to successful delivery of software products is to achieve the right level of quality processes. It is not the researcher's intention to detail a mapping between processes, software development types and their related target markets. However, at this point it is worth making the observation that the level of quality processes applied to the development of a software product is driven by a number of complex factors. At a surface level these factors could be considered in terms of type of application e.g. general purpose such as desktop publishing or safety critical such as the control rod system in a nuclear power station. However, Voas (2008) proposes that the following factors determine the level of quality or more practically, the number and type of processes required to achieve a software deliverable that is “fit for purpose” and meets predefined acceptable cost constraints:
• Definition and understanding of the environment in which the software will be deployed. i.e. are the inputs and outputs limited and therefore exhaustive test cases can be written and executed, or, are the environments not predictable and hence the factors further down this list are not fully attainable?

• Definition and understanding of the non-functional requirements such as performance, scalability, sustainability, reliability, safety, security, fault tolerance etc. A software product will need a combination of these, but not necessarily all of them. For example, a software failure in a system may cause a related system to enter an unsafe mode, this might be a minor issue in general purpose software, but is a major failure in a safety critical system.

• Definition and understanding of negative requirements. i.e. what the system must not do. The definition of negative requirements is a major element in safety critical and security systems.

• Understanding of the trade-off of resources, time and costs associated with achieving the non-functional and negative requirements. A major driving factor for this is the environment in which the software product will be deployed.

Taking the above more pragmatic definition of software quality, it can be concluded that the application of stringent quality processes could lead to an unacceptably high cost of development, which may mean the product misses its optimum release window or never achieves an acceptable ROI, but this depends on the target market environment and required quality level.

However, Jones (1995) describes an analysis carried out within IBM that concluded that software projects with high quality levels consistently met their budget and schedules, whilst projects with lower quality processes often ran late and over budget. Therefore, the approach to ‘tailoring’ software development processes to meet the needs of specific projects requires considerable
experience on the part of the project manager and/or development manager to ensure that the appropriate quality procedures are in place and followed.

Nevertheless, not all processes in the proposed framework are required for every project. For example a new product development will require a different approach to a business case than a simple upgrade project. However, the development methodology upon which the framework is built must provide a mechanism for excluding processes that are not applicable. These excluded processes must be documented as such within the projects documentation such that the processes that have been omitted are noted together with a justification for their omission as this is a basic requirement for any external audit process for example ISO 9003 TickIT.

As stated by Futrell et al. (2002), it should also be noted that each quality process must also be open to continuous improvement by applying statistical measures and acting on feedback from project members and users of the delivered software products.

### 2.2.5 Why is reuse a good thing?

Intuitively the ability to reuse previously developed software would seem to be sensible and cost effective. Given that reuse is one of the cornerstones of the Service Orientated Architecture (SOA) approach, it is reasonable to assume that this is the case.

However, if we look back to the late 80s and early 90s, object oriented analysis and design OOA/D made the same promise and on the whole failed to deliver (Schmidt, 2006). (Morisio et. al, 2002). The root causes of OOA/D not delivering reuse were many and varied from sociological issues to technical issues such as poor communication, technologies between developed components and granularity of developed objects (Schmidt, 2006).

The big step forward in SOA is not in the concept, which is not dissimilar to that of OOA/D, but in the enabling technologies that provide the ability to easily create multi-granular fully decoupled software components (known as services in SOA)
(Marks & Bell, 2006). These components can communicate using SOAP (Simple Object Access Protocol) using document format (XML) over many communication protocols such as HTTP. This coupled with the widespread use of the internet and new enterprise service bus technologies provide the perfect environment for the realisation of standalone reusable software components (Marks & Bell, 2006).

A detailed cost-benefit analysis on software reuse carried out by Rothenberger et-al (2002), concluded that software reuse does indeed save total development costs and improves productivity. However, the study also indicates that achieving reuse requires a strong reuse programme to be put in place and an acceptance of the following:

- In the early stages of introducing a programme of reuse there will be increased costs and development time required to put in place the environment and for staff to learn how to develop reusable software components

- Central service repositories are required, which need to be managed as they have an optimum size, above which additional services do not create a significant saving in reuse. This statement conflicts with experience at Celesio which has shown that reuse of components drops off for a mature product line, but accelerates when a new product range is under development and actively being enhanced, as the services required are new and do not necessarily exist in the repository build from previous software product lines.

- Reuse benefits are greater from more complex components

The above would lead us to conclude that software reuse is a good thing. However, Jones (1995) indicates, based on experiences in the American Department of Defence, that the following two items are critical in making reuse successful:
That reuse requires that the components or artefacts reused must be defect-free. Note the use of the term artefacts - reuse includes all software artefacts not just code.

That a central repository capable of handing heterogeneous data types and having excellent browsing capabilities is required. Basically, all artefacts that are approved and published for reuse must be discoverable, accessible and well documented if they are going to be reused by other developers.

This implied that any common framework must provide either its own central repository, or a link in to a specialist service repository product, in order to support a reuse approach. This also supports framework design principle 5 described in Chapter 5.

However, it is clear that any development group must first use a basic framework with fundamental development process that is capable of generating quality code before they can move on to a programme of reuse.

2.2.6 Why is a supporting tool set necessary?

From the researcher's experience at Fisher-Rosemount, the introduction of a web based software development framework resulted in processes that were previously 'paper based' being used fully. Prior to the introduction of the web based development framework, the quality processes documentation was largely ignored for the following reasons:

- Process documentation was not accessible online by the IT staff at their desks and hence they tended to be read once and then ignored, or old versions of the documents were being used. The result was misapplication of the processes and ignorance of subsequent changes to the process.
- The presentation format of the documents tended to be in the style of a QA manual and not a user guide.
• There was no online project document management application and hence developers tended to put all documentation into a limited number of design documents or as extended comments in the source code.

Introduction of a web based document management application, coupled with 'online' user guide versions of the key elements from the software process documentation, resulted in adoption of the processes by both the developers and project managers. This supports framework design principle 10 in Chapter 5.

This success was the basis on which the researcher launched the concept for the new framework, with the first success measurement being the accreditation of the Fisher-Rosemount IT department by ISO 9003 TickIT.

The above is supported by Archibald (2003) & Timmons (2000), who also indicate the following further benefits from a web based solution:

- 24 hour availability of current project information and the project document repository
- Ease of updating and exchanging current project information from any geographic location
- Improved reporting capabilities and timeliness of information
- Improved project baseline control
- Ability to build virtual teams of people located anywhere in the world
- Simplified storage and retrieval of vendor information and documents
- Ability to create a virtual project turnover/completion (punch) list
- Accelerated reaction to changes in risk, schedules, cost, or other factors
- Enhanced ability to capitalize on opportunities for schedule, cost, or other improvements

2.2.7 Why is communication a key factor?

If we consider any of the standard project management methodologies such as PRINCE2 (The Stationery Office, 2002), then the project manager is responsible for communications with project team members, stakeholders, vendors etc.
Project managers must have the correct tools (Levin, 2005) in order to communicate quickly and effectively.

Turman and McMakin (2005) indicated that web based tools facilitate task-specific feedback, notification of future tasks, priorities and collection of status information about projects. In addition, web based tools also enable virtual teams to work on single projects.

2.2.8 What key processes need to be supported by the tool-set?

There are a significant body of published work that highlight failures in the processes associated with the production of software. However, part of a classic report from the Standish Group (2004) "The Chaos Report, 1994" has been included in order to highlight the main areas in which projects fail and those key factors, when applied correctly, result in successful software deliveries. However, it should be noted that the Chaos report has been updated in 1996, 1998, 2000, and 2002 showing progress improvement in software delivery successes. Whilst the report has not been discredited, Jorgensen and Molokken (2006) have suggested that as the detailed method and results have not been published by the Standish Group, care must be taken when interpreting the published failure statistics.

The objectives of the Chaos Report were as follows:

- The scope of software project failures
- The major factors that cause software projects to fail
- The key ingredients that can reduce project failures

The Standish Group carried out a survey that included 365 U.S companies representing 8,380 software applications. The companies that responded to the survey included large, medium and small companies across all major industries.

For the analysis, software application projects were classified into three categories:
- Project success: The project is completed on-time and on-budget, with all features and functions as initially specified.

- Project challenged: The project is completed and operational but over-budget, over the time estimate, and offers fewer features and functions than originally specified.

- Project impaired: The project is cancelled at some point during the development cycle.

The overall analysis concluded that the success rate for applications was only 16.2%, while challenged projects accounted for 52.7%, and impaired (cancelled) for 31.1%. Tables 2.1, 2.2 and 2.3 capture the key factors associated with the three analysis categories.

Table 2.1 Project Success Factors

<table>
<thead>
<tr>
<th>Project Success Factors</th>
<th>% of Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. User Involvement</td>
<td>15.9%</td>
</tr>
<tr>
<td>2. Executive Management Support</td>
<td>13.9%</td>
</tr>
<tr>
<td>3. Clear Statement of Requirements</td>
<td>13.0%</td>
</tr>
<tr>
<td>4. Proper Planning</td>
<td>9.6%</td>
</tr>
<tr>
<td>5. Realistic Expectations</td>
<td>8.2%</td>
</tr>
<tr>
<td>6. Smaller Project Milestones</td>
<td>7.7%</td>
</tr>
<tr>
<td>7. Competent Staff</td>
<td>7.2%</td>
</tr>
<tr>
<td>8. Ownership</td>
<td>5.3%</td>
</tr>
<tr>
<td>9. Clear Vision &amp; Objectives</td>
<td>2.9%</td>
</tr>
</tbody>
</table>

Table 2.2 Project Challenged Factors

<table>
<thead>
<tr>
<th>Project Challenged Factors</th>
<th>% of Responses</th>
</tr>
</thead>
</table>

Released Version 1.0 Page 34 of 422
1. Lack of User Input 12.8%
2. Incomplete Requirements & Specifications 12.3%
3. Changing Requirements & Specifications 11.8%
4. Lack of Executive Support 7.5%
5. Technology Incompetence 7.0%
6. Lack of Resources 6.4%
7. Unrealistic Expectations 5.9%

Table 2.3 Project Impaired Factors

<table>
<thead>
<tr>
<th>Project Impaired Factors</th>
<th>% of Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Incomplete Requirements</td>
<td>13.1%</td>
</tr>
<tr>
<td>2. Lack of User Involvement</td>
<td>12.4%</td>
</tr>
<tr>
<td>3. Lack of Resources</td>
<td>10.6%</td>
</tr>
<tr>
<td>4. Unrealistic Expectations</td>
<td>9.9%</td>
</tr>
<tr>
<td>5. Lack of Executive Support</td>
<td>9.3%</td>
</tr>
<tr>
<td>6. Changing Requirements &amp; Specifications</td>
<td>8.7%</td>
</tr>
<tr>
<td>7. Lack of Planning</td>
<td>8.1%</td>
</tr>
<tr>
<td>8. Didn't Need It Any Longer</td>
<td>7.5%</td>
</tr>
<tr>
<td>9. Lack of IT Management</td>
<td>6.2%</td>
</tr>
<tr>
<td>10. Technology Illiteracy</td>
<td>4.3%</td>
</tr>
<tr>
<td>Other</td>
<td>9.9%</td>
</tr>
</tbody>
</table>

The Standish group used the factors in Table 2.1 to evaluate a number of successful and unsuccessful projects in order to validate these key indicator
factors. The above factors match the researcher’s experience concerning success and failure factors found in many projects (O’Neill and Dawson, 1998) and have been considered in the framework definition.

2.3 Examination of existing methodologies

A review of current literature shows that the state of the art approaches to software development fall in to three types:

- Agile methodologies
- Process assessment frameworks
- Heavy development approaches

Examples of agile methodologies are:

- Extreme Programming (XP) (Beck, 2000)
- Scrum™ (Schwaber, 2002)
- Dynamic Systems Development Method™ (DSDM) (Stepleton, 1998)
- Crystal™ (Cockburn, 2000)
- Adaptive Development™ (Highsmith, 2000)

Examples of process assessment frameworks are:

- Software Engineering Institute (SEI)
- Capability Maturity Models (SEI CMM & SEI CMM1) & ISO / IEV 15504

Examples of heavy development approaches are:

- Rational Unified Process (Kruchten, 1999)
- Department of Defence standard 2167a (DOD-STD-2167a) (Rigby, 2003)
In order to compare these methods a process map is used (see Figures 2.1, 2.2 and 2.3). Each of the above are compared using a two dimensional process map, which takes as it's Y-axis, 'Inflexible' to 'Agile' and as it's X-axis, 'High Number of Artefacts' to 'Low Number of Artefacts' as follows:

- On the X-axis, 'Low Number of Artefacts' indicates a small amount of supporting documentation and little formalisation in working procedures.
- On the X-axis, 'High Number of Artefacts' indicates comprehensive supporting documentation, traceability and configuration management supported by change control boards, design review, etc.
- On the Y-axis, 'Agile' indicates an approach that takes as its basis either an iterative or an incremental view of the development cycle. This type of approach allows for the correct emphasis to be put on architecture design and progressive requirements definition. In an iterative approach, each lifecycle phase can be executed more than once. In an incremental approach, each lifecycle phase can be executed more than once, but can also have different cycles executed in parallel streams.
- On the Y-axis, 'Inflexible' indicates the degree that the method follows the waterfall approach. Whilst the waterfall approach may be fine for enhancements to stable products, it is considered as inflexible in the situation where new software products are being developed (Knott and Dawson, 1999).

2.3.1 Agile Methodologies

According to Nawrocki et al. (2001) agile development methodologies are becoming increasingly popular especially in small development teams. Adoption of Agile development methods means that the project has to decide which artefacts will not be produced. This sacrifice is in return for flexibility and the ability to adapt to rapidly changing business environments. The approaches place greater emphasis on producing working software, rather than traceable documentation. Kruchten, (1999) confirms this, stating that rather than working to
strict requirements and development plans, an agile methodology 'responds to changes that occur during the processes.

XP (Beck, 2000), Scrum™ (Schwaber, 2003), Crystal™ (Cockburn, 2005) and Adaptive Development™ (Highsmith, 2000) grew in popularity in the late 1990s, but are built on best practices that have been around for many years, such as iterative development, continuous integration and a strong focus on executable code. One of the Agile movements strong contributions have been in the acceptance of “process” and “best practices” by developers that previously saw this as added bureaucracy. Figure 2.1 shows where agile processes fit within the process map.

![Agile Processes Diagram](image)

**Figure 2.1: The Position of Agile Processes on the Process Map**

All agile processes are placed in the lower left hand quadrant, as they are all iterative, producing low amounts of quality artefacts. Given that these processes are relatively young and unproven and do not yet provide much guidance on how to apply them practically, many organisations struggle when trying to apply these approaches (Nawrocki et al, 2001). Some of these processes are now being applied to larger projects that are more complex and hence the number of artefacts and controls required are increasing (Nawrocki et al, 2002). Hence, the diagram shows these processes moving towards the right lower quadrant.
However, Paulk, (2001) states that “only small to medium sized teams building software with vague or rapidly changing requirements” can make use of this type of methodology.

2.3.2 Process Assessment Frameworks

The next sets of processes are those aimed at quality accreditation and result in high levels of quality artefacts. These include software assessment frameworks such as SEI CMM (SEI Standards, 2004), SEI CMMI (Caputo K, 1998), ISO/IEC and ISO 9000-3 TickIT (ISO 9003, 1991). These frameworks are driven out of the need for business to ensure that software deliverables are predictable in terms of costs with no schedule overruns. As a result companies are adopting quality standards based on these process assessment frameworks in recognition of the importance of well documented software development processes.

Whilst standards documents such as The TickIT Guide (BSI, 1994) show the typical nature of these frameworks, the following is a brief summary of their characteristics:

- Version control of all project management artefacts
- Emphasis on requirements traceability through the lifecycle
- Strict change control mechanisms
- Quality audit process, including inspections and milestone deliverable reviews

Approaches that produce high numbers of quality artefacts are appropriate for complex software development projects involving large teams and possibly multiple development sites as these assessment processes produce high quality and easy to maintain software products. However, the cost of production is typically higher and time to market slower than with the more agile processes (Highsmith, 2002). If time to market is the primary objective, these processes may lead to lower quality, because there is just not sufficient time to carry out what the process requires and hence developers can ‘cut corners’.
The researcher has seen the situation where a number of projects that have been executed in an environment that has good development processes, fail due to pressure on time to market and hence a reduction in quality. This consideration has been recognised by Huang and Boehm (2006).

One example is the LinkScripts II project, which was a new store based dispensing system in Celesio AG. This was a totally new product designed to operate in store on a windows platform using C++ and PowerBuilder™. This organisation has good development methods and a solid Project Development Procedure (PDP), introduced under the management of the researcher.

The first mistake was a marketing view that the new product should be deployed within a period of just six months. Following an extensive requirements capture phase six months was impractical. However, the marketing push continued with a result that the project never really got past the requirements phase during the first six months. Using the Celesio AG (PDP), the project was restarted with realistic timescales and the correct resource profile. As the project was approaching integration testing, Marketing again stepped in and demanded that a working system be given to a list of external pharmacist users. On the face of it this looks like good practice, in getting the system in front of the users as soon as possible in order to gain usability feedback before release. The problem was that the users continued to feedback to Marketing a continuous stream of enhancements, which were not in the original project requirements. Marketing insisted that the development team carry out the requested enhancements.

The results was a further six month delay in the project and, by now, supporting the old system deployed in the stores was starting to cause issues as Marketing also insisted that the new enhancements be made to the old system in parallel. The PDP is quite clear in terms of the testing phases, defect levels, user documentation, transfer to operations and training etc., which must be completed before a product is released to the market. However, pressure to cut timescales by Marketing and now senior management resulted in short cuts being taken and the release of an incomplete and only partially tested product. This experience
supports framework design principle 1 in chapter 5. This was followed inevitably by two immediate follow-up bug releases but, of course, the product attracted a poor reputation from day one with the users and, due to not following the PDP, the timescales were extended and the project cost almost a third more than planned.

There are many examples of the quality verses time to market conflict in the published literature (Austin, 2001).

2.3.2.1 SEI CMM & CMMI: Process Assessment Framework

The Capability Maturity Model (SEI, 2010) developed by the Software Engineering Institute is designed to enable an organisation to understand and measure the quality of their software development processes. It is an assessment framework, which enables an organisation to assess its development maturity against a scale of 1 to 5. However, CMM does not tell you how to develop software, which is a mistake that is often made when the framework is adopted. This is confirmed by Reifer (2000) who notes that in a number of cases, the model is used to try to introduce processes that the model indicates are missing, but provides no indication on how this should be done in practice. This provides a strong indication that framework design principle 3 in Chapter 5 should be considered in the new framework.

CMM does provide an exhaustive list of all the possible artefacts that ‘could’ be produced as part of a software development. This can be taken as the ultimate goal as organisations embark on a mission to achieve Level 5 certification. However, Bach (1994) notes that this can result in a software development process carrying a significant amount of overhead with cumbersome processes that render the development process ineffective and make it very costly.

According to Royce (2002) another undesirable effect of CMM is that it encourages the adoption of the waterfall method rather than iterative development by not identifying issues through early and continuous integration and testing. Instead, it defines deliverable artefacts from each stage of the lifecycle.
In order to address these issues, the SEI introduced SEI CMMI (SEI Standards, 2004), which accommodates latest best practices such as iterative development. Rather than promoting the idea that, the goal should be to adopt all quality artefacts it now recommends that organisations focus on those processes that meet the organisations business objectives and mitigates development risks. This provides a strong indication that framework design principle 1 in Chapter 5 should be considered in the new framework.

2.3.2.2 SEI/IEC 15504 & ISO 9000-3 TickIT

ISO/IE 15504 (http://www.isospice.com/categories/ISO%7B47%7DIEC-15504-Standard/) is a framework similar to CMMI that assesses the maturity of software development processes on a scale of 1 to 6. The framework was derived from the Software Process Improvement and Capability Determination (SPICE) project (Ernam, 1998). This framework recommends that you only focus on those processes that are of most value to the organisation.

ISO 9000-3 TickIT is again an assessment framework against which the existence and maturity of a software processes can be assessed. Qualified TickIT auditors, on an annual basis, carry out assessments against the standard, to ensure that the process compliance is being maintained and improved upon.

ISO 9000-3TickIT is one of the worst assessment frameworks and a significant body of work exists to support this view. Seddon (1997) summarises the main negative impacts of adopting ISO standards as follows:

- ISO 900 encourages organisations to act in ways, which make things worse for their customers. For example, customers will receive a service that is defined by procedures, which is responsible for many poor service experiences.

- Quality by inspection is not quality. Inspections have a number of shortcomings, one of which is the scramble to ‘convert to the right ways to do things’ into the ISO way, just prior to an ISO inspection.
ISO 9000 starts from the flawed presumption that work is best controlled by specifying and controlling procedures. ISO encourages control of people's behaviour through inspection of their adherence to procedures. The purpose of the ISO documentation is to establish "what we say we do" and to allow the ISO audit process to determine "whether we do it". This means that the ISO approach is all about controlling output. Therefore, quality is defined as the conformance to the standard, rather than quality being all about improvement. This provides a strong indication that framework design principle 9 in Chapter 5 should be considered in the new framework.

- **The typical method of implementation is bound to cause sub-optimisation of performance.** The standard ISO approach is to look at the existing business processes, determine the gap between them and the standard, and then develop a plan to close the gap. The focus of the company becomes one of achieving the plan rather than improving the performance of the company by introducing better processes in order to hit the typical goals of reduced costs, maintenance of customer base and expansion into new markets.

- **The standard relies too much on people and particularly assessors', interpretation of quality.** The route cause of this problem is that the ISO standards do not provide specific guidance on how the standards should be implemented. This provides a strong indication that framework design principle 9 in Chapter 5 should be considered in the new framework.

- **The standard promotes, encourages, and explicitly demands actions, which cause sub-optimisation.** In order to motivate people to deliver their best performance they need to be in control. Having an external body, such as ISO, takes away that control and hence impacts on motivation and hence performance.

- **When people are subjected to external controls, they will be inclined to pay attention to only those things, which are affected by the**
controls. In short, organisations only tend to do what is required in order to pass the audit.

- **ISO 9000 has discouraged managers from learning about the theory of variation.** In summary, ISO takes the view that working to standard procedures will reduce variation. However, improvements in performance are made by using those variations that do occur in order to drive improvements.

Whilst not all of the above may apply to every organisation that adopts ISO, the researcher can find no quantitative published analysis that demonstrates that ISO by itself provides a substantial improvement to a company’s performance in its market place.

Figure 2.2 shows the assessment frameworks on the process map.

![Assessment Frameworks Processes](image)

**Figure 2.2: The Assessment Frameworks on the Process Map**

The CMM and ISO 9000-3 assessment frameworks have been placed in the upper right hand quadrant as they exhibit high levels of inflexibility and production of quality artefacts.

CMMI and ISO 15504 are trying to adopt a more flexible approach and hence have been put in the lower right hand quadrant.
2.3.3 DOD-STD, MIL-STD & RUP Processes

The American Department of Defence created a number of quality standards aimed at minimising costs and schedule slippage. These include: DOD-STD-2167, DOD-STD_2167A, MIL-STD_1521B and MIL-STD_498. However, Newberry (1995) points out that while the DOD standards initially tried to avoid the waterfall methodology, they are normally combined with MIL-STD-1521B, which requires a sequence of formal milestone reviews for requirements, design etc that are expensive to carry out and inevitably lead to a waterfall approach. MIL-STD-498 is more flexible in terms of an iterative approach, but does not really address how this could be achieved. Therefore these standards appear in the upper right hand quadrant of the process map.

The Rational Unified Process (RUP) (Kruchten, 1999) could appear almost anywhere on the process map, as it is customisable. In other words users can tailor the number of artefacts to suit their projects and use either an iterative or waterfall approach.

Rational Software has been promoting iterative development with a strong focus on executable code for many years (Booch, 1996; Kruchten, 1996).

However, experience using the RUP process on a number of projects in Celesio has highlighted the following issues:

- RUP and its artefacts cover almost the whole of the development lifecycle, but basically it does not provide guidance on when to use the artefacts and when the artefacts can be safely ignored.

- The method defines the following phases in a typical RUP project (Kruchten, 1999):
  - Inception Phase (requirements capture and project planning, risk mitigation strategies, etc).
  - Elaboration Phase (This is essentially the definition of the technical architecture design. This is a very good practice such that the
technical architecture is proven early on in the project. Countless projects have failed due to poor architecture design).

- Construction Phase (This is the development, test and alpha release trials).
- Transition Phase (Pre-release testing, rollout release preparation, user feedback changes, installation and rollout).

- Whilst the method is iterative the above phases mean that development teams can easily be forced in to a waterfall method. For example, the approach is that the requirements are mostly defined and approved during the inception phase. What Celesio then found was that whilst a number of parallel construction phases were run, the inception and elaboration phase were only executed once. This defeats the whole idea of iterative development.

- The phases in RUP should be treated as iterative, with any number being executed in parallel and any phase being executed in parallel. That is the sprit of RUP, but the mechanics of the method and the texts that describe RUP do not support the user in achieving this goal. Rather RUP is executed as a series of waterfall processes (Campbell, 2002).

- The method is not sufficiently prescriptive and hence is open to far too much interpretation, which leads to confusion and poor execution of the methodology (Campbell, 2002). (This supports framework design principle 3 in Chapter 5.)

- Often organisations try to marry RUP with a waterfall project management approach such as PRINCE2, this is possible, but all too often RUP is turned in to a waterfall method in order to map into the project management milestone review points (Larman, et al 2001). This was highlighted in a major project undertaken by the wholesale division of Celesio AG. The project was reviewed as part of the researcher's governance role and it was found that the project had adopted the
Rational Unified Process phases: Business Modelling, Requirements, Analysis & Design, Implementation, Test, and Deployment. However, each phase had a single pass through it and therefore was effectively a waterfall and not an iterative approach. The impact of this was that it took the project six months to move from the requirements to design phase as the European businesses could not agree on common business requirements for all use cases. If they had used the process iteratively, then they would have planned for the progressive definition and refinement of requirements and been able to move forward. As it turned out, the project expended significant costs in having expensive external software contractors sitting at their desks doing little or no productive work. This supports framework design principles 5 and 7 in Chapter 5.

- Another good practice of RUP is to focus on getting executable code early. Unfortunately, developers use this as an excuse to go to code early, bypassing Use Case analysis and then cut the artefacts to the bone.

- Ambler (2001) states that RUP is a single project development process, failing to address the need for enterprise level, multi-project support.

RUP, therefore, provides a strong set of development good practices and supporting tools, but does not provide strong guidance on their application (Hesse, 2001)

Figure 2.3 shows the military standards and RUP on the process map.
2.4 Literature Review Conclusion

Software development process methodologies seem to fall in to three major groups:

1. The first is the assessment framework against which current and future capabilities in software development are periodically measured, with no real guidance on how to improve software development, but rather measured against the number and quality of prescribed artefacts delivered. Zeineddine (2005) states "An ISO 9000 certification is no indicator of the quality of software products, processes or the overall quality system".

2. The second is the iterative development, low artefact methodologies that are yet to prove themselves in major software developments. Whilst it is possible to take the view that the new methodologies such as XP are aimed at rapid development in leading edge technologies, they do not enable the most efficient contributor to rapid development, that of reuse.
In order to get lowest cost and quickest time to market, reuse of previously developed components and services is critical. However, in practical terms it is more complex to design reusable components and services and even more difficult to ensure that developers 'discover' that they exist and actually use them. To do this takes strong design records, a central repository and components / services must be published and easily accessible. None of the 'agile' methods can provide this level of control.

3. The third is the heavy-artefact; prescriptive methodologies coming out of the U.S. Department of Defence and very much aimed at very large software system projects. The RUP methodology is included in this category, as its artefacts are extensive, and if followed fully, would create a very costly overhead. However, it can be applied selectively and does have an associated tools set. Nevertheless, in terms of its application, it is too complex, with little help for the context it should be applied in and it concentrates more on the development of best practice rather than providing pragmatic guidelines.

The literature review would suggest the need for a methodology which provides the following:

- Provide an adaptive framework that can be used standalone or can be merged with any of the waterfall or iterative development methodologies. (See framework design principle 5 in Chapter 5).

- Cover all aspects of an organisations software development i.e. (See framework design principles 2 and 6 in Chapter 5).
  - Integrate with the company objectives
  - Require a business case
  - Provide for clear project objectives
  - Provide for standard project documentation
  - Provide a mechanism for good project communications
- Provide for detailed customer requirements capture
- Provide processes for stakeholder management
- Provide for detailed project planning, monitoring and correction
- Provide processes for managing the introduction of deliverables
- Provide integration with various iterative development methods
- Provide processes for test management
- Provide processes for problem resolution and risk management
- Provide processes for quality process improvements

- Have programme / project management and development / delivery, quality and measurement of quality improvement

- Provide adaptability for any size of project and target quality level. (See framework design principle 4 in Chapter 5).

- Contain sufficient description of its application to guide the user and not be ambiguous. (See framework design principle 3 in Chapter 5).

- Provide a flexible and supportive tool set, which encourages both development and management staff to use it as an interactive part of their daily development activities. (See framework design principle 10 in Chapter 5).

- Support the SOA approach if this is the development methodology selected. (See framework design principle 5 in Chapter 5)

This literature review has been carried out in a proactive way combining a review of published views on software development methodologies with a comparison of the state of the art development methodologies available. In some cases the findings of the review have been illustrated by examples from the researcher's own experience at Fisher-Rosemount and Celesio AG. It is clear that none of the published methodologies satisfies all the above requirements. The RUP, perhaps, comes closest as it is a flexible system that can be tailored to different
sized projects and different quality requirements, but its lack of guidance for application still leaves it inadequate. Plus, it requires the purchase of its own specific tooling. The published methodologies are based on either quality standards, pure development level or on pure project management, but what is required is a flexible framework that encompasses all of these as well as development best practices.

The following chapters show the development and application of a proposed new software development framework that attempts to satisfy all the design principles identified in Chapter 5.
3 Chapter 3 - Research methodology

This chapter is aimed at reviewing the many different approaches that are available to a researcher and to pick the most appropriate one to achieve the thesis objectives as defined in Section 1.3. Selection of the correct research approach will have a major impact on the way the objectives are fulfilled.

3.1 Introduction

The following sections describe a number of possible research philosophies that could be adopted and explains why 'pragmatism' was found to be the most appropriate to use as a basis for this research. Having established pragmatism as the research methodology, a short review of available research approaches and strategies are included together with their suitability for inclusion in this research. Further details are provided on the selected research approach and strategies used in the execution of this research.

Sections are also included that describe how the research was conducted together with a discussion on the limitations of research approaches and strategies used.

The chapter concludes with an overall thesis roadmap that shows the sequence of the research activities and how each chapter in the thesis addresses and fulfils the ten major design principles of this research as established in Section 4.5.

3.2 Research Philosophy

A research philosophy will determine the way in which data are collected, collated, analysed and presented. If the purpose of research is to transform things that are believed to the true (doxology) in to things that are known to be true (epistemology), then the two major research philosophies that can be identified are positivist (scientific) and anti-positivist (interpretivist) (Galliers, 1992).

However, given that the research deliverables are aimed at providing practitioners of software engineering with a new framework that can be applied in total, or in part, towards improving the delivery of software projects, then a third
philosophy known as Pragmatism (Peirce, 1931; James, 1907; Dewey, 1931; Mead, 1938) will also be considered.

3.2.1 The Positivist Approach

In a positivist approach the assumption is that all knowledge can be obtained through the observation or experience of real world events. Knowledge gained through this process is entirely objective, and is essentially time invariant. It is for this reason that the positivist approach has found favour in the sciences, such as mathematics, physics, chemistry and engineering. However, there is debate concerning the applicability of a positivist approach in terms of Information Systems research, with some authors suggesting a mixed approach to the applied methodologies (Kuhn, 1960; Bjørn-Andersen, 1985; Remenyi and Williams, 1996).

3.2.2 The Anti-Positivist Approach (Interpretivism)

The pure positivist approach has been questioned from a philosophical perspective by the much cited work of Burrell and Morgan (1979), who refer to a group of thinkers that disagree with the basic assumptions of positivism, and describe this approach as 'anti-positivism.' Anti-positivism falls into the interpretative category of research approaches. Interpretism takes the position that our knowledge of reality is a social construction by human actors. Therefore, true data cannot be produced since researchers use their own preconceptions in order to construct the research enquiry and interact with the subject actors. This changes the perceptions of both the researcher and the subject actors.

Interpretism is in contrast to positivism, which makes the assumption that true data collected by a researcher can be used to test earlier hypotheses and theories.

There is a proposition that an interpretive research approach is suitable for research concerning information systems. Boland (1979, 1985) uses phenomenology and hermeneutics as the philosophical basis for his Information Systems research. Research using an interpretative approach is further
supported by Winograd and Flores (1986). They would seem to take their basis from Heidegger (1962) who argued that the separation of subject and object prevents the unity of “being in-the-world”. Winograd and Flores reached the conclusion that a new research approach is needed for Information Systems research with action as the main focus. The notion of action is expanded in the next section concerning Pragmatism.

Two further papers by Orlikowski tend to support the interpretative approach for Information Systems research and, in particular, research involving organisational process change associated with the introduction of CASE (Computer Aided Software Engineering) tools. Orlikowski and Robey (1991) examine the extent to which information technology deployed in work processes can facilitate changes in processes and organisations. Orlikowski (1992) draws on ‘structuration’ theory (Giddens, 1984), and provides an interpretation of the stages in the use of software tools from the user perspective. Both of these papers address similar research issues as those to be considered in this thesis.

3.2.3 The Pragmatic Approach (Pragmatism)

Pragmatism has been seen as an alternative to both positivism and anti-positivism. In certain perspectives, pragmatism shares the objections made by a hermeneutic and constructivistic anti-positivism. Pragmatism objects to an over-emphasis of subjective interpretations. It is not sufficient to say that an interpretation makes sense; it must make sense practically.

A foundational precept of pragmatism is that the meaning of an idea or a concept is the practical consequences of the idea/concept. The meaning of it is the different actions, which are carried out, based on the belief in this concept. Peirce (1878) formulated this pragmatic principle: “Thus, we come down to what is tangible and practical as the root of every real distinction, no matter how subtle it might be; and there is no distinction of meaning so fine as to consist in anything but a possible difference of practice”. In other words pragmatism encompasses a pragmatic view of the meaning of concepts and ideas (Rescher, 2000).
Interpretivist approaches would seem to have their roots in social-constructivist thinking (Walsham, 1995). "The aim of interpretive research is to understand how members of a social group, through their participation in social processes, enact their particular realities and endow them meaning, and to show how these meanings, beliefs and intentions of the members help to constitute their social action" (Orlikowski & Baroudi, 1991). In a pragmatic approach these aims can be agreed to a certain extent. Interpretivist approaches focus on the subjective ("meanings, beliefs and intentions"). Rescher (2000) describes the pragmatic position as the following: "In the human realm, praxis (doing) has primacy over theoria (understanding) because all understanding must itself be the product of doing: whatever we know (understand) is the product of inquiry, an activity of ours". In a pragmatic approach, emphasis is put on the significance of actions and the external world of material artefacts and our interaction with these through interventions.

There would appear to be a tendency in scientific research to abstract and generalise which can often led to conceptualisations, which can be difficult to understand. A pragmatic approach indicates the necessity to make abstractions with practical application in the real world. However, pragmatism is not totally against abstraction and conceptualisation, it seems to consider this to be fundamental within the human thought process. However, pragmatism warns against conceptualisations, which are not clearly grounded in the empirical and real practical world.

A pragmatist researcher can make observations and, based on these, generalise, but at the same time be aware of possible applications of the developed knowledge, and thus try to apply the knowledge in a way that makes it useful for practical application. The pragmatist researcher is interested in what differences this knowledge will have in practice and tries to translate knowledge into action. A pragmatist position can explain and justify the widespread application of action research and methods development in information systems research. Both these research strategies are aiming at formulating and trying out what would be better to practice.
The overall research aim in this thesis is to improve knowledge derived from initial theories by formulating them practically and applying them in the real world of software engineering. The results derived from taking these research actions are aimed at contributing to improvements in software engineering processes and practices. The ultimate goal is to disseminate the knowledge gained in a practical and directly applicable way to both the academic community and software engineering practitioners. Pragmatism has therefore been chosen as the philosophy, as it seems more appropriate for this type of action based research. A positivist philosophy was ruled out on the basis that every environment in which the proposed new framework would be applied would be different and hence the adoption approach would vary. In addition, it should be noted that the researcher will be involved at many levels in the research and hence will influence both the enquiry and observation, which effectively excludes positivism. Whilst an interpretivist philosophy would seem to be a closer fit, it lacks the core motivation that all enquiry and observation must be grounded in practical applicability.

3.3 Research approach

3.3.1 Research environment
The research approach to use is influenced to a large extent by the academic and industrial settings in which the research work is carried out (Orlikowski and Baroudi, 1991). Furthermore, the research work reported in this thesis has been carried out to a large extent within the industrial information systems setting in an action research oriented fashion complemented by case studies and limited use of online surveys. These case studies were carried out both to evolve the proposed concepts presented in this thesis and to obtain evaluations of their validity in an industrial information systems environment. A consequence of this approach is that the IT processes under research involve many additional factors that have an influence both on the research work and the behaviour during the actual case studies and subsequent evaluation. The research work involves many roles and skills among the participating people,
information systems, commercial drivers and logistic aspects, existing methods and tools, etc. Some of the actors are central to the research work performed, whereas others are more peripheral, although they may exercise influence on the research work in different ways and on different levels.

Many disciplines in science are relevant, such as management science and social science, in addition to the main subject area of computer science. Therefore, a pragmatic approach has been chosen as the overall research philosophy for this thesis and, due to the nature of the holistic framework forming the core element of the research objective, a limited number of the available research approaches have been selected and are presented in this chapter.

3.3.2 Research approaches considered

There are many different combinations of research approaches that could have been adopted and used as a framework to undertake the research. The researcher has considered three broad styles of research approach. Livari (1991) provides a summary of the approaches as follows:

- Constructive Research Methods, including:
  - Conceptual development
  - Technical development
- Nomothetic Research Methods, including:
  - Formal mathematical analysis
  - Experiments; laboratory and field
  - Field studies and surveys
- Idiographic Research Methods, including:
  - Action research
  - Case studies

Each of the approaches is considered briefly as follows:

The constructive approach is concerned with developing frameworks, refining concepts or pursuing technical developments. The approach allows models and frameworks to be created that do not describe any existing reality or do not
necessarily have any "physical" realisation (Cornford and Smithson, 1996). The use of this approach in research aimed at defining a practical (physical) software development framework would seem to indicate that the constructive research approach would not be applicable.

Definitions of both nomothetic and idiographic research can be found by livar (1991). Nomothetic research is based on a search for, and evidence to support, general laws or theories that will cover a whole range of cases. This type of research places emphasises on repeatable hypothesis testing within a fixed environment and, hence, is closer to a positivist research philosophy. Given the environment variability faced in the software industry, creation of a very abstracted development framework would have minimal value in a one software development environment. In addition, the approach is not in line with the pragmatic approach chosen for this research.

The idiographic research approach is aimed towards the observation of particular processes, cases and events in order to capture and analyse them within their natural (real world) environment. This approach is the basis of both interpretivism and pragmatism.

An idiographic approach takes as its basis the analysis of subjective observations based on participation or remote observation with real world events (Cornford and Smithson, 1996). Given the nature of the research this would seem to be the most appropriate approach to employ. The research is contains both participation by the researcher and detailed observation, monitoring and analysis of repeated application of the emerging framework and, as such, is definitely a pragmatic philosophy.

3.3.3 Research strategies considered

There are a number of papers published addressing the many approaches to research strategies (Galliers, 1992; Baskerville and Wood-Harper, 1996; Alavi, 1994; Benbasat et al, 1987; Remenyi and Williams, 1996). However, based on Järvinen's (2001) hierarchical structure for the classification of research strategies, their applicability in the context of this research is discussed as follows.

- Mathematical research strategies
• This attempts to prove or disprove a theorem or assertion. Whilst there are some theorems within the research areas, such as the use of polynomial equations in test completion predictability, this is not a major part of the main research approach and therefore this strategy will not be used.

• Conceptual analytical strategies
  • This would seem to be the analysis of existing theories, models and frameworks from previous studies in order to create a generalised conclusion. As can be seen from the literature review in Chapter Two, software development frameworks have been considered and the learning is factored in to the research. However, the intention is not to create a generalised result based on previously defined frameworks, but to develop a new framework based on repeated and rigorous application of the evolving framework in a real world context. Therefore, this strategy will not be used as part of this research.

• Theory testing
  • This uses a number of methods such as laboratory experiments, specialist types of case study (Lee, 1989), field studies and field testing. This strategy could be used for part of the research as it includes case studies and testing of theories in practice, but does not cover the theory creation that is required in generating a new framework and associated methodologies.

• Theory creating
  • There are a number of theory creating strategies:
    - Contextualism (Pettigrew, 1985)
    - Grounded theory (Strauss and Corbin, 1990)
    - Action research (Wallen, 1996)
    - Case studies (Yin, 1989)
    - Participant observation (Smith, 1978)
    - Phenomenological studies (Langdridge, 2007)
    - Ethographic methods (Hammersley, 1990)
Theory creating strategies would seem to be a better fit for this research and in particular: Contextualism, Grounded Theory, Action Research and Case Studies. The following provides a short overview of each strategy that is thought to be a candidate for inclusion in this research and its applicability to this research context.

3.3.3.1 Contextualism
Pettigrew (1985) indicates that one of the main requirements of a contextualist analysis is to understand the emergent, situational, and holistic features of an organism or process within its context, rather than prepare a model of only a sub-set of the variables. Pettigrew (1985) suggests that researchers proposing to use contextualist based approaches should place themselves into an organisation, use multiple information gathering techniques, and act as a consultant, principally by giving feedback to the organisation. Given that the researcher will be part of the organisations in which the proposed framework is developed and that a good proportion of data will be qualitative in nature, then an element of the contextual approach will be included in the research strategy.

3.3.3.2 Grounded theory
Grounded theory is a practical approach to the analysis of qualitative data. Glaser and Strauss (1967) indicate that research is conducted in different environments, by researchers who will impose different levels of personal and professional qualities on the research. They further argue that it would not be feasible to impose a single set of methodological rules to be followed in all cases. In addition, they indicate that by doing so would reduce the quality of the research. Strauss (1987) suggests that the analysis of qualitative data must be aimed at the generation of new concepts and theories. Glaser and Strauss (1967) and Strauss (1987) highlight that theories must be 'grounded' in an empirical reality, providing an iterative loop for theory, checking and refinement (Denscombe, 2002). Glaser and Strauss (1967) also state the requirement for researchers to start with an open mind. On initial inspection it would appear that grounded theory would be applicable as research will have a substantial number of variables to consider, some of which are not known at the outset. However, given the environments in which the framework is to be developed and
validated are not all 'green field' and the researcher has substantial theoretical and practical experience in the research subject, then fully applying grounded theory with no preconceptions would not be possible. Therefore grounded theory will be used as part of the research strategy, but with the constraint as described.

3.3.3.3 Case studies

There are several examples of the use of case methodology in the literature. Yin (1993) listed several examples along with the appropriate research design in each case. A criticism of case study methodology is that its dependence on a single case renders it incapable of providing a generalising conclusion. Yin (1993) presented Giddens' view that considered case methodology "microscopic" because it "lacked a sufficient number" of cases. Hamel et al. (1993) and Yin (1984, 1989, 1993, 1994) argued that the relative size of the sample does not transform a multiple case into a macroscopic study. The goal of the study should establish the parameters, and then should be applied to all research. In this way, even a single case could be considered acceptable, provided it met the established objective.

Yin (1989) indicated that general applicability results from the set of methodological qualities of the case, and the rigor with which the case is constructed. Yin detailed the procedures that would satisfy the required methodological rigor. Case study can be seen to satisfy the three tenets of the qualitative method: describing, understanding, and explaining.

The body of literature in case study research is "primitive and limited" (Yin, 1994), when compared to that of experimental or quasi-experimental research. The requirements and inflexibility of the latter forms of research can make case studies the only viable alternative. It is a fact that case studies do not need to have a minimum number of cases, or to randomly "select" cases. The researcher must work with the real world situation that presents itself. This specific research will use case studies to evaluate both processes and outcomes and therefore will include both qualitative and quantitative data (Evans, 1976; Gopelrud, 1989) where appropriate.

Yin (1994) identified five elements of research design that are necessary for case studies:
- A study's questions
- Its propositions, if any
- Its unit(s) of analysis
- The logic linking the data to the propositions
- The criteria for interpreting the findings (Yin, 1994, p. 20).

The research questions are likely to be "how" and "why" type questions and their definition is the initial task of the researcher. The research propositions sometimes derive from the "how" and "why" questions. An exploratory study, rather than having propositions, would have a stated purpose or criteria on which the success will be measured. The unit of analysis defines what the case is. This could be groups, organisations or countries, but it is the primary unit of analysis. Linking the data to propositions and the criteria for interpreting the findings are the least developed aspects in case studies (Yin, 1994).

Construct validity is a problem in case study research. This has been a source of criticism because of the potential subjectivity or the investigator. Yin (1994) proposed three ways in which to counteract this:

- Multiple sources of evidence can establish a chain of evidence, and a draft case study report should be reviewed by key informants.
- Internal validity is a concern only in causal (explanatory) cases. This is usually a problem of "inferences" in case studies, and can be dealt with using pattern-matching.
- External validity deals with knowing whether the results can be generalised beyond the immediate case. Some of the criticism against case studies in this area relate to single-case studies. However, that criticism is directed at the statistical and not the analytical generalisation that is the basis of case studies. Reliability is achieved in many ways in a case study. One of the most important methods is the development of the case study protocol.
Yin (1994) presented the protocol as a major component in asserting the reliability of the case study research. A typical protocol should have the following sections:

- An overview of the case study project (objectives, issues, topics being investigated)
- Field procedures (credentials and access to sites, sources of information)
- Case study questions (specific questions that the investigator must keep in mind during data collection)
- A guide for case study report (outline, format for the narrative) (Yin, 1994, p. 64).

Stake (1995), and Yin (1994) identified six sources of evidence in case studies. The following is not an ordered list, but tries to reflect the research of both Yin (1994) and Stake (1995):

- Documents
- Archival records
- Interviews
- Direct observation
- Participant-observation
- Physical artefacts

Documents on the scope of this research can be requirements, design, test specifications, or quality procedures, process guidelines, test results, project management documentation etc. In the interest of triangulation of evidence, the documents serve to corroborate the evidence from other sources. Documents are also useful for making inferences about events. However, documents can be
misleading in the hands of inexperienced researchers, which has been a criticism of case study research. The researcher in this case is very experienced in the field and in evaluating the validity of documents included in the case studies. Archival documents in this case are records of events from previous development projects including test results. The investigator has to be careful in evaluating the accuracy of the records before using them. Even if the records are quantitative, they might still not be accurate.

Interviews are an important source of case study information. There are several types of interview:

- Open-ended is an open-ended interview, respondents are asked to comment about certain events. They may propose solutions or provide insight into events. They may also corroborate evidence obtained from other sources.

- Focused interview is used in a situation where the respondent is interviewed for a limited period of time. This technique is often used to confirm data collected from another source.

- Structured interview is similar to a survey with the questions being detailed and developed in advance.

- Direct observation occurs when a field visit is conducted during the case study. This technique is useful for providing additional information about the topic being studied. The reliability is enhanced when more than one observer is involved in the task.

- Participant-observation makes the researcher into an active participant in the events being studied. The technique provides some unusual opportunities for collecting data, but could face some major problems as well. The researcher could well alter the course of events as part of the group, which may not be helpful to the study.
• Physical artefacts can be tools, instruments, or some other physical evidence that may be collected during the study as part of a field visit. In this case the artefacts are a combination of project management, software development documentation and delivered application software. Both classes of artefact are managed in electronic repositories.

Given that the refinement and validation of the proposed framework will be carried out by continuous application of the emerging framework in multiple real world software development projects, the researcher feels that the use of case study as part of the research strategy is appropriate.

3.3.3.4 Action Research

Action research is concerned with processes and phenomena that would not have occurred without the active intervention of the researcher or research team (Wallen, 1996). Denscombe (2002) suggests four defining characteristics of action research:

• Practical
  o Aimed at dealing with real-world problems and issues, typically in work and organisational settings

• Change
  o Change is regarded as an integral part of the research, providing both a way of dealing with practical problems, and as a means of discovering more about an event.

• Cyclical process
  o The research involves a feedback loop to itself, in which initial findings generate possibilities for future investigation.

• Participation
  o Practitioners are crucial to the research program, and their participation is active not passive.

Given that action research involves the participation of the researcher and active contribution from one or more actors there are a number of possible researcher/actor combinations and hence research action types. For example:
The action research types are described as follows:

3.3.3.5 Participatory action research:
The components of participatory action research would seem to make it somewhat different from other social research methods and seem to be more like a methodology than a method. Its qualities of being both an active research practice and one based on the principles of democracy with the relocation of authority from the researcher to the community of interest is a central tenant of the research method (McNiff, 1988). Within participatory action research, the researcher is the instrument for facilitating change, rather than the owner, director and expert in the research project, or as Whyte (1991, p.40) puts it, the researcher has the role of research coach. This means that the researched, or the subject group, are the owners and instigators of the research rather than the researcher. In the words of Rapoport (1970, p.1 cited in Robson, 1993, p.438): ‘Action research aims to contribute to both the practical concerns of people in an immediate problematic situation and to the goals of social science by joint collaboration within a mutually acceptable framework’. As such, participatory action research as a research method/methodology is characterised by the strong and active involvement and high degree of participation of those with the research objective (Whyte, 1991).

However, participatory action research has been criticised by other social researchers. These criticisms focus on how its participation, democracy and external ownership aspects can greatly reduce the validity of the research and the rigour of the methods used, and question whether participatory action research leads to sound, scientific, valid, reliable, usable research outcomes. Further criticism centres on what is perceived as a moralising tone in the methodology. For example, Adelman (1989, cited in Robson 1993,p.440) argues that participatory action research claims, of being an ‘alternative research paradigm, as a democratising force and means of
achieving informed, practical change arising from issues at the grass roots are overbearing'. The main criticism here is that participatory action research could be seen more as an ideology of how research should be carried out rather than a practical research method. The other issue is the research completion date. Unlike most other research methods, which tend to be timed undertakings with clear start and stop points, the iterations of participatory action research continue until the problem is resolved.

Given that the research in this case will place the researcher in the role of research director and as such will direct the research activities it is felt that this strategy will not be suitable for adoption in this research case.

3.3.3.6 Action Science:

Making reference to Argyris (1983) and Argyris, Putnam and Smith (1985) it would appear that a conceptual research model and associated processes, which seem to be both a theory of social systems and an intervention method, has been progressively defined over a number of years. This method seems to be appropriate to the research of organisational system dynamics based on communication flows, and interrelationships.

The basic idea of the model is that, despite their declared values, people follow undeclared rules. These undeclared rules may prevent them behaving as they might have consciously thought. This can result in interpersonal and organisational processes which contain many hidden problems and agendas. At the same time, taboos prevent the problems or their existence being brought in to the open. These undeclared rules and hidden agendas guide people's actions and interactions within an organisation.

The methodology essentially depends upon agreeing on processes which identify and deal with those undeclared rules which can prevent the honest exchange of information. The methodology places a strong emphasis on the people involved in the research being honest about their own intentions, and about their assumptions
about each other's motives i.e. the method can provide an improved set of communication processes, which could enhance other action research methods.

Argyris and Schön (1989, 1991) acknowledge action science as a form of action research but identify an important difference in focus. They indicate that normal social research is not capable of producing valid information. Without valid information the rigour of any action research may be undermined. Their papers appear to indicate that action science is a research method that obtains valid information about social systems, whereas a number of other research methods fail to do this as competently.

It is concluded that action science is an appropriate choice of method if there is strong person dynamics, especially if hidden agendas appear to be operating. However, it probably requires better interpersonal skills and a willingness to confront hidden issues than do the other action research methods. It also appears to be aimed at the analysis of social systems and behaviours. In conclusion, this methodology is not appropriate in this case.

3.3.3.7 Soft Systems Methodology:

Soft systems methodology is a non-numerical systems approach to action research. A description of the methodology can be found in numerous references (eg. Checkland, 1981, 1992; Checkland and Scholes, 1990; Davies and Ledington, 1991; and Patching, 1990). In order to describe the methodology, consider an inquiry process that consists of three dialectics. In each dialectic, the researcher alternates between two types of activity. The first activity refines the second. Figure 3.1 outlines Checkland's soft systems methodology inquiry process as a series of dialectics.
Figure 3.1: Checkland's soft systems methodology inquiry process

The following attempts to summarise each of the dialectics in Figure 3.1:

The first step is for the researcher to immerse themselves in the reality, using an approach not dissimilar to participant observation (Lofland and Lofland, 1984). During this first step the researcher periodically stands back from the situation, reflecting and trying to make sense of it. At these reflection points the researcher considers questions such as: what is the system achieving or trying to achieve? On return to the study, the researcher can check if the research process is adequately capturing the essential information from the system. This process continues until the researcher is content with the description of the essential system functions.

In the second step, the researcher forgets about reality, and works from the description of the essential system functions defined during the first step. An ideal system is devised that achieves the system's actual or intended functions. The researcher then moves between essence and ideal, until eventually a decision is taken that an effective way for the system to operate has been captured and described.

In the third step, a comparison between the ideal and actual is made. This comparison process may identify missing elements of the ideal, or improvements to the system.
In the fourth step, the practical and beneficial system improvements are put in to practice, forming the fourth dialectic.

It should be noted that the above process is very similar to that of the Kolb (1984) learning cycle as shown in Figure 3.3.

Soft systems methodology is iterative, providing an improved understanding through each iteration. Mismatch between each dialectic or continued uncertainty results in further iterations. In systems terminology, the ‘essence’ (see Figure 3.1) becomes the required functions (or problem definition). Checkland (1992) calls them root definitions. To check that the problem definition is adequate he proposes a ‘CATWOE’ analysis. CATWOE is an acronym defined as follows:

- **Clients** - those who directly benefit or suffer e.g. customers;
- **Actors** - the players (individuals, groups, institutions and agencies), who perform the scenes, read and interpret the script, regulate, push and improvise;
- **Transformations** - what are the transformations associated with the system that generate a product, service or process change? How are they achieved?
- **Weltanschauung or world-view** - what is going on in the wider world that is influencing and shaping the "situation" and need for the system to adapt?
- **Owners** - the activity is ultimately "controlled" or paid for by owners. Who are they and what are their imperatives? How do they exercise their ownership power?
- **Environment** - the trends, events and demands of the political, legal, economic, social, demographic, technological, ethical, competitive, natural environments provide the context for the situation and specific problem arena.

The CATWOE analysis helps in working out a "root definition" and expressing the domain of the problem and helps avoid early conclusions.
The 'ideal' is also defined in systems terms by proposing an ideal way of transforming the inputs into outputs. System modelling can be used to propose ways in which the goals of the researched system can be achieved. Checkland (1981) described this system modelling as a seven-step process. The steps are as follows:

1. the problem unstructured (examination of the problem situation)
2. the problem expressed (analysis of the problem and its constituent elements)
3. root definitions of relevant systems (identification of the key facets of the system)
4. conceptual models (capture of relevant conceptual models)
5. compare the expressed problem to the conceptual models (comparison of conceptual/ ideal to actual)
6. feasible and desirable change (define and selection of feasible options)
7. action to improve the problem situation (put in to practice proposed changes)

Soft systems methodology is well suited to the analysis of information systems and also decision making systems in general (Loo and Lee, 2001).

Checkland's soft systems methodology appears to be a response to difficulties in applying more positivist based research methodologies to business and decision (human activity system) problems. Positivist methodologies tend to emphasise:

- measurable and objective criteria
- the isolation and control over variables
- top-down decomposition of systems into sub-systems.

It is, therefore, concluded that, given the iterative nature of the proposed framework, which involves extensive problem definition, human interaction and interpretation, modelling and multiple implementation and refinement cycles in different environments, soft system methodology is more suited than a more rigorous positivist
methodology. In addition, it is recognised that in this particular project that the process of analysis (human interaction) is as important as precision in the data and outcomes, and also that the application of soft systems methodology will by its application affect change in the organisations in which it is applied. However, by engaging in multiple organisations and affecting change the overall ideal model can be progressively and practically refined.

3.3.3.8 Evaluation:

Evaluation would seem to contain a mixture of methodologies with approaches that vary from those which are very positivist in their nature (Suchman, 1967) to those which are anti-positivist (Guba and Lincoln, 1989). Cook and Shadish (1986) summarise the progressive movement over time from a positivist to anti-positivist position and associated reasons. A number of different approaches to the evaluation methodology can be found in the literature (e.g. Patton, 1990; Guba, 1990; Guba and Lincoln, 1981, 1989; Lincoln and Guba, 1985).

However, in order to come to a conclusion concerning the appropriateness of the methodology in terms of this work the evaluation model was considered as proposed originally by Snyder (cited by Dick, 2003). It would seem that the Snyder model is aimed at the creation of a self improving system. The approach is based on a systems evaluation model devised by Snyder. The major elements of the model are:

- Resources
- Activities
- Immediate effects
- Targets
- Ideals

In essence, resources are consumed by activities which produce immediate effects in the achievement of defined and agreed targets. Successful realisation of the targets then contributes to the progressive move to the ideal system model. According to
Dick (2003) “The operation of a project is analysed in these terms — by identifying resources, activities, immediate effects, targets and ideals, and the way they interact. The information provided by this analysis is then used to:

- understand how the project operates, and so improve its operation;
- understand how well the project operates, and so communicate this to funding bodies, directors, and others;
- build in processes for ongoing monitoring, and so continue to improve project functioning.

In short, the overall process allows the functioning of the project to be understood and improved and demonstrated, and for the improvement to be ongoing. This is for the benefit of the project team, its clientele, and others with an interest in it.” In other words a project or system is analysed with pre-set targets (goals) for improvement, from which a series of project or system improvements are derived. These improvements being assigned to the team involved in the analysis to enact.

It is concluded that this methodology is really very similar to TQM (total quality management) (Boaden, 1997). It seems to make the assumption that a process is already in place and functioning which then requires an improvement process to be applied in order to bring it towards an ideally functioning process. The extent of the process improvement is set as targets assigned to the team involved in the analysis of the project or system. This assumes that the process is already in existence and mature as this would be a prerequisite for being able to set reasonable targets for its improvement. In this research case it is assumed that a significant portion of processes are not pre-existent and that refinement of processes will be a future part of continuous improvement and hence this will not form part of this initial research.

Denscombe (2002) provides a good summary of the advantages and disadvantages of action research in general. Action research would seem not to specify constraints by which information is to be gathered nor does it specify the analysis to be performed. It would appear to apply a strong element of pragmatism in its approach, allowing a variety of different data and analyses to be performed (Denscombe, 2002; Susman and Evered, 1978). It is true to say that the theoretical field of action
research tends to be oriented towards research on social systems and related behaviours; it can therefore be considered to be a supportive framework for this research. The applicability derives primarily from the fact that the research work has been performed within two very different company settings and applied to numerous software development projects containing many multi-disciplined teams.

As the researcher participated in a significant portion of the research, either directly, or indirectly, action research is a good framework to use for this research as the researcher influences and studies the course of events in an organisation or in society in general. The result of action research or subsequent application does not always generate successful results created through the intervention. However, provided that the intervention has led to new knowledge and that the reasons for the less successful results have been understood, the results of the action research will still be important factor in to the ongoing research.

3.4 Conducting the research

Based on the review of the research approaches and strategies included previously in this chapter, the most suitable way forward for this research is to adopt the research philosophy of ‘pragmatism’, together with an idiographic approach and using a combination of strategies: Contextualism, Grounded Theory, Action Research and Case Studies. As the researcher will fulfil an active role in the definition, validation and continuous evolution of the intended framework, either directly, or indirectly, action research and case study are considered a sound framework to use for this research.

Whilst the management of the two companies in which the research was carried out were aware that this was taking place, the participants in the research were mostly unaware. Whilst recognising that this could be an ethical issue, it was balanced against the possible obtrusive influences that would result in the participant's awareness of the research. With this in mind, the research has been divided in to two main phases as discussed in the next section.
3.4.1 Definition of phases

A two-stage approach was planned in order to create and validate the proposed framework against the objectives as defined in Section 1.3.

- In the plan for the first stage, which covered objectives 1, 2 and parts of 5, the need for such improved software development methodology would be demonstrated and the requirements determined by real case study application to a number of real projects. This would be supplemented by reference to published literature, together with the repeated application and refinement of the emerging framework by continued observation and monitoring by the researcher on many subsequent software development projects. The researcher would not be directly involved in a significant number of the software development projects to which the framework was subsequently applied. This phase also includes the development of an on-line repository tool, which is used to support the framework.

- In the second stage, which covered objectives 3, 4 and the completion of 5, the framework would be further applied to a significant number of software development projects of ever increasing size and complexity and in a more commercial and fast moving business environment. This would result in the framework being refined further and significantly extended in order to address the challenges of both development and of quality audit requirements. In addition, a second case study would be applied, designed along the lines of the first case study, which was also applied to the design and implementation of a new software product. This provided a good benchmark from measure improvements between the initial core framework and the later extended framework.

The initial development of the framework and its verification on 'real' business critical projects would be carried out at Fisher-Rosemount Systems (FRS). Fisher-Rosemount is a highly technically orientated company with a good track record of adoption of quality standards. It has the benefit that both business and customers have an engineering background, which means that both the business users and
customers understand issues such as requirements capture, testing and quality control in general.

After the introduction of the new framework into the software development organisation in FRS, the next challenge would be to test its scalability and applicability in a much more commercial and less quality aware organisation in the second stage of the research. An ideal opportunity to carry out this stage of the research became available when the author changed employer in a move to Celesio AG. The scalability and applicability could then be 'stress tested' by introducing the framework into the UK divisions of Celesio AG.

Celesio AG is the largest Wholesale & Retail Pharmaceutical group in Europe, with companies in sixteen countries. From a cultural perspective the company had very little understanding of the need for quality standards and, at best, a business view whose ROI (return on investment) horizon was six months. The business culture was to achieve double-digit growth each year in each country, with no constraints on how that was to be achieved.

The UK division consisted of the UK wholesaler, AAH and the Lloyds Pharmacy Retail chain. These two companies were acquired by Celesio in 1997 and hence were a newly merged organisation when the author started the introduction of the framework in 1998.

The problem statement for Celesio AG and indeed the challenges that the framework had to meet would be substantial. The merged companies had created a single IT department to support both sides of the business. However, the individual companies had moved from the North West of England to the Midlands, but over 85% of the IT staff had failed to relocate and had left the respective companies. In addition, the legacy systems both companies operated were up to 25 years old, and in a number of cases, duplicated. For example, both AAH and Lloyds had their own warehouse management and pharmacy dispensing systems. The IT department had no development methodology, no configuration management, or release testing, and no documentation. In fact they just carried out ad-hoc code changes at the direct request of business users. The business users had no understanding of how to provide
requirements, why software took time to develop and no grasp of quality. Therefore the relationship between the business and IT was very confrontational.

To size the software problem, the UK division had in the past developed all its business applications in-house and now had a team of 240 staff supporting 94 separate software products. Clearly, introduction of the framework would be a significant challenge and would refine the framework in order to operate in a more commercial and fast pace environment.

3.5 Research roadmap

Figure 3.2 provides an overview of the main research activities mapped against each chapter in the thesis together with an indication of which activities contribute to the ten framework design principles as defined in Section 4.5

<table>
<thead>
<tr>
<th>Thesis Chapter</th>
<th>Research Stages &amp; Activities</th>
<th>Design Principles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter One</td>
<td></td>
<td>Principle One</td>
</tr>
<tr>
<td>Chapter Two</td>
<td></td>
<td>Principle Two</td>
</tr>
<tr>
<td>Chapter Four</td>
<td></td>
<td>Principle Three</td>
</tr>
<tr>
<td>Chapter Five</td>
<td></td>
<td>Principle Four</td>
</tr>
<tr>
<td>Chapter Six</td>
<td></td>
<td>Principle Five</td>
</tr>
<tr>
<td>Chapter Seven</td>
<td></td>
<td>Principle Six</td>
</tr>
<tr>
<td>Chapter Eight</td>
<td></td>
<td>Principle Seven</td>
</tr>
<tr>
<td>Chapter Nine</td>
<td></td>
<td>Principle Eight</td>
</tr>
<tr>
<td>Chapter Ten</td>
<td></td>
<td>Principle Nine</td>
</tr>
<tr>
<td>Chapter Eleven</td>
<td></td>
<td>Principle Ten</td>
</tr>
</tbody>
</table>

Figure 3.2: Thesis Roadmap

Key to figure:
- 'Chapter' refers to the main chapters contributing to the creation of the framework
- 'Principle' refers to the framework design principles
- 'Stage' refers to the grouping of the research activities
• 'Case study' refers to the four major case studies carried out as part of the framework validation. Three are labelled on the roadmap the fourth being a combination of the Process Performance and Process Capability elements.

• 'Core framework' refers to the central processes within the framework, but excludes process performance, process capability and integration with industry standard methodologies.

• 'Verification on projects' refers to the repeated use of the framework in multiple real software development projects using a structured case study approach.

• 'On-line repository' is a electronic web based repository on which the framework is made available and acts as the master version controlled source of all framework artefacts.

• 'PRINCE2' (2002) is an industry standard project management methodology.

• 'RUP' (2007) is an industry standard software development methodology.

3.6 Conclusions

This chapter has reviewed the research philosophies, approaches and strategies available, selecting those that were felt to be most appropriate for this specific research case. The execution of the research in a number of information systems companies, in a real world context, implies that the researcher and the experiences of the researcher in each company will play a role in the research outcome. The research approaches and strategies, therefore, form a basis for structured reflection upon the results and associated influences. Through the application of structured research strategies, a considered analysis of the results and generalised formulations can be proposed and to a certain extent be validated by experimentation in the industrial company setting in an action research oriented and case study based approach. A framework that describes this approach is the basic learning cycle (the Lewinian Experiential Learning Model, see Figure 3.3) described by Kolb (1984).
Concrete Experience
(doing / having an experience)

Active Experimentation
(planning / trying out what you have learned)

Reflective Observation
(reviewing / reflecting on the experience)

Abstract Conceptualisation
(concluding / learning from the experience)

Figure 3.3: A basic learning cycle (adapted from Kolb, 1984).

In terms of this thesis, the starting point for experience is derived from published information (refer to Chapter Two) on the success or failure factors associated with either an overall software project, or specific, or groups of specific, software engineering processes. Observations and reflections that are made on the basis of the knowledge obtained and experience led to the formation of new or improved process concepts and framework generalisations. These new or improved process concepts and framework generalisations led to new or revised approaches. The new approach, its process concepts, and its framework generalisations with the corresponding impacts are formulated and then tested in new real-world situations through active experimentation on real software development projects. Continued application of the process concepts and framework generalisations again led to further knowledge acquisition and hence further refinements of the framework.

This approach of a basic cycle for knowledge generation can be given structure in order to provide rigor through the application of additional scientific standard research approaches, and strategies at different phases along the basic learning cycle. A number of such approaches and strategies are described in the chapter, which are
based on the assumption that the 'experience' element of the Kolb (1984) learning cycle is actually practical real world experimentation in the form of action research and case study.

The chapter indicates that the research is split in two separate stages. The first stage takes an initial framework as the basis for the first case study where the application of the framework improves the software development processes and deliverables, when compared to previous software developments within the same product line. The results are then analysed in order to verify that the framework has made improvements and to identify where further improvements to the framework are required. Given the selected research approach, the framework can only really be fully established by the continuous application and results analysis based on a significant number of different software development projects. This is the approach taken by the researcher over a two year period. Towards the end of the stage a second case study was carried out, again on a new software product development, in order to validate the now much extended framework.

However, using the framework in one specific company environment does not fully validate the framework for wider use in other companies and industries. Therefore the framework was introduced into a very different business environment, which was, to all intents and purposes, a green field site in a multi-site business environment. This presents the challenge both in terms of multiple development teams, varying levels of expected quality, and much shorter timescales across the various sites. The aim was to test the flexibility, scalability and robustness of the framework. This resulted in further refinement of the framework and some significant extensions in order to introduce auditable quality levels in to the various development sites, without the need for expensive ISO preparation initiatives, and to enable the framework to integrate with existing industry standard methodologies that may already be in place and working.

Finally, the chapter contains a roadmap that provides an overview of each stage together with an indication of how each chapter contributes to the fulfilment the framework design principles as defined in Section 4.5.
Chapter 4 - Are Current Quality Standards Sufficient For Quality Software?

This chapter considers if today's quality standards are sufficient to enable the delivery of high quality, on time, on budget software products with a high level of customer satisfaction.

Case studies carried out by the researcher whilst working for Fisher-Rosemount and GEC Power Instrumentation & Control are used to illustrate that software quality standards alone are not sufficient. The thesis contains four major longitudinal (Kung and Zeger, 1986) case studies, two within this Chapter: Case Study One at Fisher-Rosemount, Case Study Two at GEC Power Instrumentation. The third, Case Study Three, at Fisher-Rosemount is described in Chapter 7, with details in Appendix G. The fourth, Case Study Four, again at Fisher-Rosemount, is described in Chapter 8.

The methodology chapter indicates that case study forms part of the overall thesis research approach. The case study is considered by Benbasat et al (1987) to be viable for three reasons:

1. It is necessary to study the phenomenon in its natural setting.
2. The researcher can ask "how" and "why" questions, so as to understand the nature and complexity of the processes taking place.
3. Research is being conducted in an area where few, if any, previous studies have been undertaken.

Case studies can be single or multiple designs, where a multiple design follows a replication rather than sampling logic. Case studies One and Two can be considered as single, whilst case studies Three and Four can be considered as a multiple design as they are structured in the same way as Case Study One and are applied to a very similar environment being again a new software product creation within the same company and development department.

Stake (1995), and Yin (1994) identified at least six sources of evidence in case studies.

1. Documents
2. Archival records
3. Interviews
4. Direct observation
5. Participant-observation
6. Physical artefacts

Case studies One and Two involved direct observation, whilst the third involved participant-observation in that the researcher designed the case study, played a role in the case study project and carried out the analysis, but in all case studies used peer reviews within the department in order to validate the observations. A more general set of case study observations were used by the researcher to analysis results from a significant number of additional software projects, using direct observation in order to establish more evidence of the general applicability of the emerging framework. The overall results for these additional case studies are tabulated in Tables: 5.1, 9.1, 9.3, 10.3 & 10.11.

By using a mixture of participant observation, direct observation and peer reviews, the researcher has endeavoured to minimise any direct influences on the results.

Details on each case study and the limitations encountered can be found in Appendix G.

The conclusions drawn from Case Study One and Two in this chapter, the literature review chapter and further papers such as the extensive questionnaire carried out by White and Fortune (2002), which shows that over half the companies in the survey have had to create their own customised methodology for software production, are used to create an initial set of design principles for a new framework.

4.1 Introduction

There is a significant body of work that highlights countless software project failures. For example:
• A study (Sauer et al, 2003) carried out by Oxford University & Computer Weekly indicated that only 16% of all software projects in the U.K were considered to be fully successful.

• A study (Taylor, 2001) carried out by the British Computer Society concluded that only 3 out of 500 software development projects surveyed met the assessments criteria for success.

• A report (Standish Group, 2003) published by the Standish Group estimates a success rate of around 34% for software development projects in America. It also indicates that this is an improvement on a similar survey carried out by them in 1995, which concluded that only 16% of projects were successful. This is also broadly supported by Jorgensen and Molokken (2006)

Some of them are high profile, whilst others go unnoticed. The unnoticed ones can be covered by the now cultural statements ‘the software is always late’ or ‘there’s a fault, must be the software’. It seems that industry accepts huge software development costs and the inevitability of delays. Texts are now produced containing details on software product failures (Jones, 1995). Even casual observation of today’s major IT projects, such as the National Health Services programme for IT, shows some major project mistakes that could drive that initiative into software legend (Wilkinson, 2006), that would amount to around £20 Billion of taxpayer money. For example, in the NHS IT programme it can be observed that contracts for the networks, enterprise integration products, centralised patient medical records, hosting, hardware and applications, have been distributed between many different suppliers.

Requirements for the NHS systems are being generated by various steering committees. Recent indications are that the users (British Medical Association, Hospitals and Pharmacies) have not been consulted in terms of the requirements and associated impacts. (NHS Connecting for Health Web-Site, 2005). Hence the user base has started to reject the programme.

It has been common practice in the programme to award supply contracts to suppliers before the requirements are defined and agreed with the users i.e. to agree
and assign contracts to suppliers, which are not based on the final scope or agreed requirements with the users. This approach is not uncommon in the programme and has been used to try and show early progress. The researcher has seen this occur several times during his time with Celesio AG when the business sponsor decides to bypass the IT department and purchase a product directly from the market, without carrying out a proper selection process, which critically includes the detailed matching of functional and non-functional requirements. Often this is a result of poor alignment between business and IT (Papp, 1998; King, 1995; Henderson and Venkatraman, 1996; Earl, 1993; Luftman, Lewis and Oldach, 1993; Luftman 1996; Goff, 1993; Liebs, 1992; Watson and Brancheau, 1991).

The following key list of typical major programme issues is based on the researcher's own experience of running and reviewing major change programmes in both Fisher-Rosemount and Celesio and from various publications (e.g. Boehm, 1991). The researcher has also been involved in the implementation of pharmacy systems associated with the NHS IT programme. However, these types of key project and programme success criteria are highlighted in a number of project and programme management methodology communications such as the Central Computer and Telecommunications Agency (CCTA) publication (CCTA, 1999).

- Suppliers should be minimised such that there is only one organisation responsible for delivery and operations as the chances of successful final integration is exponentially increased by every additional supplier that is involved
- Generation of requirements needs to be done quickly and with those users who understand the practical benefits
- It is necessary to consult the users before generating the requirements and bring them in to the programme as active participants

The above list, coupled with the continued reports in the computer media, indicates that the NHS programme has failed to address its approach to the provision of IT systems, within a complex, multi-project programme in the correct way. This assertion is further supported by the recent recognition by the NpflT (now renamed...
Connecting for Health) programme director general that the programme needs more "clinical engagement" (Connecting for Health Web-Site, 2005).

The list can be extended (Whittaker, 1999), but it is clear that the mistakes of the past and even good practice are often ignored and hence can have significant impacts on major national investments. This is further supported a call for an independent audit of the Connecting for health programme (Collins, 2005).

Do software projects fail because they do not use recognised standards? To answer this question it is first necessary to understand the difference between a quality standard and a quality framework. Often these are used in the same context and they are quite different.

Two well-known quality standards are ISO 9000-3 TickIT (The TickIT Guide, January 2001, Issue 5) and the Capability Maturity Model (CMM) (Caputo, 1998). ISO 9000-3 is an extension of the ISO 9000 standard, which itself was derived from the earlier BS 5750 standard. CMM was generated from the US military need to improve the standard and consistency of software as a component of larger defence systems. Both of these standards outline general guidelines and principles for the specification, development and testing of software products.

The ISO standard is aimed at attaining a general level of software quality that enables repeatable performance and quality. The CMM model provides five levels of improvement, each of which requires either the introduction of components of quality such as configuration management or standard software process practices such as peer reviews (Doolan, 1992). In order to pass each level a tick list of capability is provided by either self assessment or by formal assessment by trained SEI (Software Engineering Institute) assessors.

Both these standards suffer from a lack of detailed information on how they should be applied. For example, no supporting tools or templates are available. This leads to interpretation and inevitably patchy application of the standards. In the case of CMM there are a number of publications that provide implementation guides in order to adopt each quality standard (Chrissis et al., 2003). In fact Caputo (1998) states about CMM “The assumptions that support effective software process improvements are
hidden in the CMM. The CMM does not state the assumptions directly, but you can discover them if you know where and how to look for them.

Caputo goes on to describe how it took a number of years to identify the basic assumptions that underpin all CMM process areas, based on his own software development experience, in order to simplify the CMM model such that it could be applied effectively. If this is the case, then CMM is open to interpretation and hence can have inconsistencies in its application.

In addition, the standards do not address in adequate detail two further key areas:

- Project management
- Development methodology

This is where the difference between a quality standard and a quality framework can be understood.

The researcher recognised that quality standards alone are not sufficient and that a new approach is necessary. However, any new approach should not simply add to existing quality standards and invent its own project management and development methodology and supporting tools. Rather it needs to provide a framework in which a number of project management and development approaches can be integrated and managed within the common framework in such a way that they act in a complementary way and within an easily managed and transparent environment.

For example, a widely used project management methodology is PRINCE2 (Stationery Office, 2002). The U.K government developed the PRINCE2 method initially in order that projects could be managed using a consistent approach. This method has now been taken over by various training organizations and developed to include both project and programme management components (http://www.maventraining.co.uk/prince2).

Unlike a standard, a framework should provide detailed training courses and be supported by standard templates for management deliverables e.g. Project Initiation Documents, Exception Reports, Risk Logs etc. However, this is a general project
management methodology for the management of not only software, but also any type of project.

Given that PRINCE2 is well established with training courses and templates, a quality framework should be able to adopt the methodology, rather than try to reinvent the wheel. The fundamental development assumption of such project management methodologies is that the process is based on a ‘waterfall’ sequence (Knott and Dawson, 1999). Where companies appear to run into difficulties is when they then try and adopt a software development methodology based on an iterative approach (Knott and Dawson, 1999). Trying to integrate a waterfall and iterative approach can result in the project management trying to monitor a development process that appears to have a different view on milestone deliveries.

A widely adopted example of such a development methodology is the Rational Unified Process (RUP). Application of both the PRINCE2 and RUP methods takes a great deal of experience in order to prevent a rift between the project management and the development teams, which can lead to project failure. For example, the PRINCE2 method is based on a number of fixed project phases e.g. Requirements, Design and Implementation etc. In effect the end of each phase is a deliverable in itself and is used as a major project management progress measurement milestone. However, if we consider the RUP method that uses an iterative technique that allows a project to pass through each phase more than once and has sub-projects running in different phases. The development activities can be running in a number of streams, each of which can be in a different development phase. Therefore, taking a simplistic waterfall approach to managing the project can result in low visibility in terms of the actual development progress. A framework is needed to provide an effective bridge between the largely waterfall project management and the new iterative development methodologies.

What we appear to have available today is, at the top-level, very generalised quality standards and a whole array of methodologies at the detailed level, that can result in very poor application of the overall software development process. However, there is no easy way to bridge the gap, which leaves a significant proportion of companies
developing their own fragmented approaches with no reference to standards or using any methodology at all. This assertion is supported by the survey carried out by White and Fortune (2002). The survey consisted of a project management method, tools and techniques usage questionnaire. The questionnaire was sent out to 995 project managers, in 620 separate organisations, within both the private and public sectors. The results indicated that the application of project management methods, tools and techniques is by no means extensive.

Table 4.1 provides a summary of the questionnaire results, which were provided by 236 respondents.

Table 4.1: Summary of Project Management Questionnaire (from White and Fortune (2002)).

<table>
<thead>
<tr>
<th>% Result</th>
<th>Abs Result</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2%</td>
<td>5</td>
<td>Respondents stated that they did not use any methods, tools or techniques.</td>
</tr>
<tr>
<td>28%</td>
<td>66</td>
<td>Respondents did not use any methodology.</td>
</tr>
<tr>
<td>95%</td>
<td>225</td>
<td>Respondents use at least one project management tool.</td>
</tr>
<tr>
<td>52%</td>
<td>123</td>
<td>Respondents did not use any decision-making techniques.</td>
</tr>
<tr>
<td>54%</td>
<td>128</td>
<td>Respondents used their own ‘in house’ project management method.</td>
</tr>
<tr>
<td>77%</td>
<td>182</td>
<td>Respondents used an ‘off the shelf’ project scheduling tool.</td>
</tr>
<tr>
<td>64%</td>
<td>152</td>
<td>Respondents used Gantt charts.</td>
</tr>
<tr>
<td>42%</td>
<td>99</td>
<td>Respondents had encountered limitations with the methods, tools and techniques used.</td>
</tr>
</tbody>
</table>

The actual project management questionnaire results can be found in Appendix D.

Two conclusions, which reinforce the need to develop a new software development framework, are:
• Over half the companies found it necessary to develop their own project management methodology.

• Almost half the project managers encountered limitations in the methods, tools and techniques currently available.

A further survey carried out by Fitzgerald (1997) observed that 60% of survey respondents were not using any formalised methodology and, in those that indicated use of a methodology, it was not fully adopted and heavily customised in order to be used in practice.

4.2 The drive for standards

There is a drive for improved standards in the software industry. Published information from ISO indicates that a significant proportion of companies have either attained ISO certification or are in the process of doing so.

There are currently many quoted drivers for a company to feel compelled to attain ISO certification. The following sections explain what these drivers are and how they are used to pressurise companies in to attaining the ISO standard in particular.

4.2.1 Customer pressure via bid list questionnaires

The Fisher-Rosemount business provides control systems to the petrochemical processing and pharmaceutical manufacturing industries. As such, the software quality has to be of the highest level as its failure could case large collateral damage. The major suppliers require all suppliers whose products have a software content to have attained at least ISO 9001 and in the case of US companies level 2 CMM.

4.2.2 Keeping up with competitors

With over 52,000 company registrations to ISO in the UK alone, management is put under pressure in terms of ensuring they do not miss opportunities based on being beaten on a bid by a company with ISO certification. This was a major driver in the Fisher-Rosemount business and in the Celesio business in preparation for the possible ruling by the NHS that all healthcare systems must be ISO certified.
4.2.3 More certain customer satisfaction

During the 90s one of the major quality measures that was often stated, but hardly ever measured, was customer satisfaction.

In Fisher-Rosemount the ISO quality standards being applied took as their basis the need to have early signed-off user requirements and zero defects at delivery. The effect was that user requirements were being interpreted by the software development teams in such a way that they made the least impact on the software product and hence reduced the likelihood of introducing defects. For example, in Fisher-Rosemount each software development project was allowed a proportion of development time to address the open defect backlog before release. However, whilst zero defects within a released product was the overall goal, in reality a certain number of defects at each classification level were set in order that the software product could be released. In theory this seems to be reasonable. However, an inspection of the defects revealed that the defects being fixed were those that had the shortest fix time and lowest risk in terms of impacting other areas of the software. Again this would seem like a reasonable risk reduction strategy. Further inspection revealed that over 50% of the remaining defects were high customer satisfaction impact defects that would improve the efficiency of the customer’s usage of the software product. Therefore, driving for low defect numbers is not the target. The target should be to delivery a software product that is as close as possible to the user’s requirements.

On the surface low defect numbers gives an improvement in terms of the defects per line of uncommented code metric, but in reality the user community in Fisher-Rosemount were not seeing the delivery of software products that were making them more productive in their business and hence the company saw a direct hit on its profitability. The reason was that the software products were not only sold to customers, but also used by the company’s internal engineering staff to deliver solutions to customers.

Customer satisfaction requires a close alignment of business people with IT staff in order to jointly develop the solution. This can be partly addressed by iterative
methodologies, but also by ensuring that the business sponsors play an active role in the development of their business solutions.

4.2.4 More efficient quality management

It can be argued that having a quality standard in place improves the quality of the software development process. However, as shown by the case study later in this chapter, following ISO based quality standards does not necessarily lead to an improvement in overall quality deliverables. The fact that there are more processes in place and that they can be monitored, is not an indication that the processes actually improve the quality of the software delivered.

4.2.5 Software quality improvement / lower defect numbers / higher defect removal rates

As the title of this section indicates the ISO measure of software quality focuses on defect rates as the major metric for software quality, as does the BS 5750 and Crosby principles that preceded it. As the case study later in this chapter shows, a focus on defect reduction can cause major issues in software delivery. The use of ISO quality standards in the case study did little to reduce defect numbers in the product.

Quality improvements should be more focussed on a balance of acceptable defect rate and levels, verses overall cost of production and customer satisfaction. For example, compare a Microsoft product in which the average defect discovery by users of one defect per week, with the UNIX operating system, which has 1 defect per year. Both are acceptable when considering the market, costs and usage of the software products as the widespread use of both shows.

4.2.6 Lowering software production costs

The obvious challenge to lowering the overall software production costs is that introducing a highly bureaucratic quality process will add a cost to the production of software. Clearly it is also necessary to consider the whole lifecycle of the software
product. In order to remain competitive software production costs need to be acceptable and this includes the support phase post release. Releasing a software product that contains a high number of defects will require help desk support, bug fix releases and perhaps a follow-up major patch release. All of which add to the total cost of the product to the company.

The post release phases are where the ISO standards are weak and little information is available on the costs associated with the software product once it is released. Therefore any quality framework must consider when a product is realistically ready for release and is not when there are no more defects discovered in testing or, even worse, when a fixed testing time has been applied. Both approaches will lead to major costs after release and poor levels of customer satisfaction. Techniques for determining the optimum time for release of a software product will be considered later in this thesis.

4.2.7 As contractual protection, where a supplier can be held to the standard if they are in breach.

This is a nice idea in theory, but in practice it cannot be proved that any standard, if followed, will result in a quality product being delivered on-time. In addition, it is seldom helpful in customer / supplier relationships to build up a defensive contract as, at the end of the day, nothing in a court of law can compensate a company for the time to market lost in a court of law. The inference that the introduction of ISO type standards will automatically result in the above benefits is somewhat simplistic. The ISO standards are based on the idea that tasks are best controlled by having a set of detailed procedures. This is not the case in the researcher's experience, as ridged procedure-based approaches result in sub-optimal processes, and actually slows down improvements to the management and development processes.

Continuous improvements should not mean improvements to the written procedures, but must be aimed at optimisation of each quality process based on a detailed understanding of the process itself i.e. the typical procedures provided by standard quality frameworks must be adapted to optimise delivery and quality for the specific market and be proven to be effective by application. The researcher experienced, in
Fisher-Rosemount, a central quality department responsible for the creation and update of software quality procedures that were not directly applicable to the software products being developed. The result was that each development team had its own set of quality processes designed to optimise the type of software that it produced. The result was a disjoint between the company standard procedures and the actual quality processes being applied. The introduction of a sound development infrastructure, coupled with a continuous improvement culture, based on treating the whole process as a delivery system, will be much more effective in moving the organisation towards the benefits listed above than adopting a published software quality framework and issuing as a set of procedures.

The above two paragraphs would seem to fly in the face of conventional wisdom. However, if we consider that applying a set of fixed software standards to the production of software which is delivered in many variants and using many, frequently changing technologies then we are assuming that one fixed set of software standards fits all software production variants.

Let us consider further the possible variables that are associated with software production:

- Required operational reliability such as commercial, military, safety critical, home market, office market, embedded etc
- Relationship between IT department or software supplier and business sponsor
- Investment level
- Maturity of IT staff
- Skill level of IT management
- Skill level of IT development staff
- Complexity of the project in terms of business delivery requirements
- Overall size of the project
- Sensitivity of the customer base
- New product
- Enhancement to existing product
- Adoption of new technology

One set of essentially static, quality standards cannot optimise software deliveries that have so many variables to consider and this indeed was the experience of Fisher-Rosemount and Celesio.

Whilst a sound understanding of quality standards is essential as a basis three further attributes are required in order to build an optimised quality framework.

1. The first is a solid understanding of software engineering principles, in order to know when and at what level quality processes should be applied in any given software construction project.

2. The second is a flexible framework that lets improvements to the software construction process be fed rapidly into an evolving quality framework.

3. The third is an agreed business aligned IT strategy and associated IT and business architecture into which the software products fit. This is a newly emerging role in large software organisations and is known as Enterprise Architecture (Schekkerman, 2008). Enterprise Architecture ensures that a single business and IT vision is maintained in order that the right projects are executed, using the right technology and at an acceptable quality/cost level.

Notice that the continuous improvement bullet is conspicuous by its absence from the ISO adoption drives as discussed in the previous sections. This really starts to highlight the general lack of understanding of what the standards should be trying to achieve.

During the researcher's time at Fisher-Rosemount, the company hosted six separate ISO audits and acted as a reference site for other companies wishing to improve their software quality levels. During the last two ISO audits the researcher interviewed the ISO auditors and the quality manager at Fisher-Rosemount. The main question asked was why they felt that so many ISO audits either resulted in failure or
generated a number of non-compliances? The response was always the same that "the standards are not at fault, but our interpretation of them". Given that the ISO standards give little guidance in how to implement the standard, this shows either a lack of understanding by the various bodies of either the complexity of software development or the level of guidance required.

Clearly, the standards are not going to change radically in the very near future. In order to bridge the gap, another more practical way needs to be provided to improve software quality. The standards can be regarded as a general guideline on quality objectives, but a comprehensive framework and programme of improvement needs to be put in place if real benefits are to be achieved.

Software standards have had poor press in a number of industries. Jones (1995) suggests that the most visible aspect of ISO certification is a significant increase in the paper / administration at the expense of the certification process itself.

The TickiT guide (ISO 9003, 1991) gives the benefits of using a ‘Quality System’ as "an improvement in quality and repeatability, which is reflected in increased process efficiencies and a reduction in failure costs". The guide shows the following cost savings, which are welcome, but all targeted after the product is released:

- Cost of post delivery defect correction
- Cost of overruns (this would be more appropriate if it was 'scheduled' overruns)
- Reduction in unnecessary high maintenance costs
- Indirect costs due to poor quality (such as loss of business due to poor quality)

All are most welcome, but the actual cost savings are hard to quantify being post release, and do not address the other main issues that impact companies: development costs and time to market. If the market window is missed than the other benefits are academic. It is little wonder that software companies are sceptical about adopting ISO and other standards (Zuckermann, 1994), as the benefits quoted are difficult to quantify. The type of data listed above is usually available in a qualitative form. This makes the decision to invest in attaining a recognised software quality standard difficult to reconcile within an organization.
A general review of the current standards themselves shows that the standards lag behind state of the art project management and development techniques. It is the researcher’s contention that standards will never achieve their objectives until they are provided within a framework that includes practical examples, templates and a way of monitoring achievement against the standard.

In summary criticisms of the ISO / SEI type standards can be summarised as follows:

- The standards are written to suit a generic quality process which makes them difficult to apply to specific applications
- The standards do not take in to account the companies business objectives
- Application of standards results in an increase in software paperwork (this is a major deterrent for most software developers)
- Standards do not necessarily improve time to market
- The typical approach to ISO implementation causes sub-optimisation of performance

4.3 Case Study One – Fisher-Rosemount

In the mid eighties Fisher-Rosemount, as it was then, was immersed in a ‘Crosby’ quality culture. Crosby (1986) asserts that all company quality procedures should say 'We will deliver defect free products to our clients, on time'. As those in the software industry will appreciate, achieving this Holy Grail is often not possible with today's customer needs and the rate at which software technology is evolving.

The first Fisher-Rosemount case study (Case Study One) in this thesis will show the negative effect this 'simple' statement had on the companies R&D development. The chronological decision making processes are shown in Appendix A - Case Study Fisher-Rosemount. It should be noted that the project was for a major new software product, an engineering configuration application for configuring process control systems, consisting of around 900,000 lines of source code (not including comments and white space).
This section examines the case study introduced in a conference paper by the researcher (O’Neill and Dawson, 1998) involving the development of a control software product by Fisher-Rosemount. The case study considers the issues associated with developing a control product using the Crosby quality approach and the BS 5750 standard. The case study is of a project that was late, over budget and produced software considered to be unsatisfactory by the users.

The researcher took the role of direct observer designing the case study structure, identifying the data sources, doing the analysis and arranging the peer reviews.

Most software professionals will be familiar with quality standards such as ISO9000-3 and TickIT, 1991 which are intended to show that an organisation has processes in place that can produce quality software. Most will also be familiar with the concept of a “quality culture” where the whole company from senior managers to the lowest level staff, focus on quality as their prime concern. This case study shows, however, that conforming to quality standards and developing a quality culture does not guarantee quality software and may even inhibit quality.

The quality standard applied in the case study was a mixture of Crosby and BS5750, a predecessor to ISO9000-3. These involved working within defined processes with systematic records kept to verify conformance to independent auditors. The company had also sent all personnel on training courses to develop the zero defect quality culture advocated by Crosby. This working environment, however, did not prevent the following problems occurring:

1. Meeting the company business objectives

The most significant problem was that there was no clear vision of how the project met the company business objectives. The objectives set at the outset of the project were necessarily vague, as the requirements had not been fully established at that point. However, one objective was clear and that was for zero defects in the software. Although there were many good points taught in the Crosby, 1985 course this was the most obvious directly measurable goal. Consequently this became the main focus of attention by senior management, taking on an unwarranted importance. This meant that the team lost its focus on
the reasons for the project's existence that of improving the performance speed of the product. The resulting product was severely criticised by the customers for being too slow and difficult to install.

If a development team loses its understanding of the requirements and objectives, then there is little chance that customer expectations will be fulfilled, as indicated by Anderson (1996). The lesson is that the overall company objectives should be known and from the outset a business case should be made for each project based on the company objectives. Once the project is underway a business case is required for each major decision especially where the allocation of resources is required.

A further lesson is that the business case must also be communicated to the development team to ensure it is the principle driving force behind the decisions taken as the project progresses. Only when the business objectives are understood at all levels can a relevant project plan be drawn up with topics such as project priorities, risk management strategies and support services properly identified. This would have enabled the zero defect philosophy to have taken a more natural place in the project priorities when the cost of pursuing the philosophy to the very last defect was compared with the cost of performance failure and of delayed delivery.

2. Lack of detailed requirements in the early stages

Part of the problem of aligning the project with the business objectives experienced in the case study was that there was a lack of marketing input in the early stages of the project. The detailed requirements of the new product were not produced until the project had been underway for eight months. Until then the developers were producing planning estimates and initial designs based on their own concept of what was needed. The project manager himself eventually needed to spend time visiting the customer sites to gather information on requirements.

The lesson shows clearly that marketing input is required at project initiation to establish the business case and senior management must be involved to ensure
the project fits into the overall company objectives. The problem of late requirements could never have occurred if a business case for the project had been prepared as the requirements would have been a necessary part of the justification for the project. Furthermore a focus on the business objectives would have shown that assigning the project manager to collect the customer requirements was not the most effective means of achieving the objectives.

3. Excessive paperwork

The project team became over burdened by the BS5750 quality standard. As many have found in the past, such quality standards can generate excessive paperwork resulting in developers spending too high a proportion of their time engaged in bureaucratic administration (Jones, 1995). This is a fault in the implementation of the standards rather than in the standards themselves. The full documentation of every last process detail is time consuming and unnecessary. However, some processes are critical for the effective development and quality of a product. A business case approach can be used to know when and where to define a process to BS5750 or any other standard. If the problems caused by not defining a process are likely to cost less than defining and recording the process then this is a valid reason for leaving it out of the implementation of the standard.

4. Delayed decision on database technology

It was six months into the project before a final decision was taken on the relational database platform for the project. This was caused by a dispute between the developers whose experience and preference was for Sybase and senior management who wanted the "company standard" Oracle software.

Once again a clear business objective could have prevented this problem. The dispute over whether the project should adopt Oracle or Sybase, or over any project alternatives could have been resolved immediately by comparing the contribution of each alternative towards the business objectives. If one would cost more in training and in programming time because of the developer inexperience, this could be compared with the company cost of utilising more than one vendor's software. The cost in such comparisons can be measured in terms of money,
effect on time scales, market position, customer satisfaction, the drain on resources or any other business objective. If there are technical differences the costs associated with the advantages and deficiencies of each should be compared.

5. No risk analysis

The system produced in the case study was an all-or-nothing replacement for an earlier version marketed by the company involving new hardware as well as software. As no thought was given to the provision of a safeguard of a fall back to the previous step if a problem occurred, the company could not deliver the new product until all aspects had been completed. This meant that when the first delivery proved unsatisfactory, new versions had to be hurriedly produced, and inevitably they too were not completely satisfactory. The lesson of this is that part of any business case should be a risk analysis identifying the risks and the consequence of each event. This analysis should have revealed the potential problems of a single, all-or-nothing delivery. Incremental developments will nearly always be justified on business grounds.

6. Unrealistic project schedule

The problem of unspecified requirements in the early stages also led to a quite unrealistic initial schedule for the project. The initial estimates showed a timeline of 15 months for 9 developers, i.e. 135 man-months. These estimates were plucked from the air by an inexperienced project manager who had not managed large software projects before, working with no real requirements available and little historical data from previous projects. Senior management were more concerned with allocating the schedule to meet their perceived market need than to support the project manager with the necessary experience. The schedule was further hampered by the late decision to present the software at a trade show, taking key personnel from the project at a critical time. The actual project took 24 months and required 595 man-months.

Again a focus on the business objectives could have prevented this problem. A realistic time schedule is necessary for a valid business case to be made,
justifying time and resources being spent on obtaining the realistic estimates required. Although there may be pressures for shortened time scales the allocation of unrealistic schedules can only reduce the chances of achieving the business objectives. Variations in the schedule should be allowed only if it can be justified in serving the business objectives. The advantages of attending the trade show should have been set against the delays in delivery caused. Only if the trade show could be shown to better serve the overall business case should it have been allowed to delay the project schedule.

7. Poor staff planning and resourcing

The initial unrealistic schedules lead to poor staff planning and resourcing. The inaccuracies of the project estimates (only a quarter of the final total effort) were not recognized until nine months in to the project. This meant that too few developers were allocated in the first instance. Later, when it became obvious that the project would not meet the target delivery dates there was a sudden increase in the allocated staff, doubling the numbers a year into the project and doubling again as the project neared its completion. The number of staff over the project duration is shown in Figure 4.1. The sudden influx of staff gave its own problems with too many to train by too few, hard pressed, existing team members. The release testing was carried out in somewhat desperation as can be seen by the project resource profile for the last four months.

![Staff Profile for Case Study](image)
The lesson of the staffing problems serves, firstly, to emphasise the need for a realistic schedule so that staff allocation can be properly planned. The business cost of panic measures such as the allocation of many new team members at the same time must be avoided. Secondly, at the end of the project, the staff resource load did not fall off gracefully, but came to an abrupt end. This indicates that the project was not ready for release at that point.

8. Inadequate testing philosophy

A written communication from the Marketing Product Manager was issued prior to the release, which stated that the product would have totally unacceptable performance for customers. This, against a backdrop of it being a productivity tool should have been a trigger to take immediate action. There was a need to establish just how unacceptable the performance was as soon as possible to generate some additional time for the problem to be addressed.

The project should have gone for an extensive field trial release (got it to the field, whilst working on the performance improvements). However, the testing philosophy proved to be inadequate. The project used a 9-week pre-product testing approach. This meant that the product was being tested while the code was still being written. There was too little testing planned for the completed software with only four beta test sites chosen. This proved to be too few and of too short duration to establish the major problems in the first software release. As the objectives for the project were too vague apart from the zero defect requirement, the beta testing concentrated on this aspect and failed to pick up some of the important performance and installation deficiencies in the product. Consequently customers were dissatisfied with the software delivered.

The lesson here is that to keep the project in line with the business objectives the project goals must be clearly stated and clearly achievable and there must be a measure of progress towards achieving the goals. These measurements must be recorded and reviewed throughout the project to keep it on track. The need for measurable project goals emphasizes the need for adequately testing. Had the project objectives been clearly stated then it would have been obvious that the
four beta sites were clearly inadequate to measure the success in meeting the objectives and that immediate action was necessary following the Marketing Product Manager's input.

In summary, most of the problems in the case study can be attributed; directly or indirectly, to a lack of clear vision about what the project was for and where it was going. The development processes in place were clearly not leading to quality software delivered on time, within budget, and with complete customer satisfaction.

The level of 'project management' methods, tools and techniques in use within Fisher-Rosemount were practically non-existent. This was also reflected in the overall level of management and technical documentation produced during a project. This was particularly surprising considering that this was a BS 5750 accredited company, using quality standards administered by a separate quality department.

4.4 Case Study Two – GEC Power Instrumentation & Control
This case study highlights an additional requirement that any new framework must address namely the impact of software development right across the business and must consider all aspects from stakeholder involvement to delivery and deployment. In order to illustrate this consider the following short case study carried out by the researcher during his employment with GEC Power Instrumentation & Control. Again the researcher acted as a direct observer using peer reviews to add validation to the results.

Zahran (1997) contends that the risks associated with software creation can be reduced by the application of a well-defined and documented software development process. Although this statement seems rather obvious now, it would not have been so obvious 7 or even 5 years earlier. At that time, the silver bullet was 'methodologies' and 'CASE Environment'. Zahran (1997) made this observation, indicating that the application of tools to a development environment which had poor software development process, resulted in little or no discernible improvement.

The researcher has had practical experience of this issue whilst working as a consultant for one of the GEC SCADA / TELECONTROL companies. During
1991/1993 the company's current software creation process, had led them to a point where software projects were inevitably late, costs were high, defect levels were high and configuration management was a major problem (poor control on the use of multiple lines of decent). The brief was to introduce a computer based design tool to support the Yourdon™ methodology. The belief at that time seemed to be that improvements in the design, and speed of design of the software would improve the overall project delivery times and hence start to redress the situation.

The introduction of 'Teamwork™', which was a software design tool, running on VAXstations did make the application of Yourdon™ and specifically the subsequent modifications to the design a lot quicker than doing it manually. However, the root of the 'Software Crisis' for the company lay not in the quality of the software design or to some extent testing, but in the overall software process. The software process is not restricted to the production of code itself, but includes all aspects of a software project.

The major weaknesses of the company, identified by the researcher were as follows:

- Development techniques were not applied consistently, i.e. not all projects used Yourdon™, even some developers within a single project did not use it. This lack of consistency was found in many areas within the project.

- Projects were always given impossible time scales (or 'challenges'- a management term given to a software project that is under resourced). The approach applied by the company was to bid for as many projects as possible, with the hope that one or two contracts would be secured. There is no issue with this approach. However, in order to close the deal, the schedules / costs were such that the software part of the project was in timescale trouble from the start.

- As a direct result of the impossible timescales, defect levels were always very high and would be addressed after the first version of the software was sent to the customer's site. The idea being that the customer site would be in commissioning phase and would not need fully working software. Whilst management operated this approach from a commercial perspective, it inevitably resulted in the development team working on a massive defect list, which often introduced a
defect for each one removed. System testing became non-existent, which was again a direct result of the impossible original schedules. There is a tendency when projects are under timescale pressure to reduce the time available for testing, which is false economy.

Therefore, improvements in the software process need to be addressed right across a company's business and at all levels. The business environment must be managed to ensure that software projects are successful; looking for a silver bullet tool or methodology alone, will not guarantee success.

4.4.1 Lessons learned from the Case Studies

The most significant lesson to be learned from the Fisher-Rosemount Case Study is that the company must not lose sight of its business objectives at any point during a project. The overall company objectives should be known and a business case made for each project based on the company objectives. Once the project is underway a business case is required for each major decision especially where the allocation of resources is required.

Marketing input is required at the project initiation to establish the business case and senior management must be involved to ensure the project fits into the overall company objectives. The business case must also be communicated to the development team to ensure it is the principle driving force behind the decisions taken as the project progresses. Only when the business objectives are understood at all levels can a relevant project plan be drawn up with project priorities, risk management strategies and support services properly identified.

There is a danger that any project team can lose its way, the earlier this occurs in a project, and the harder it is to recover from. In order to keep the project in line with the business objectives the project goals must be clearly stated and clearly achievable. This implies there must be a measure of progress towards achieving the goals. These measurements must be recorded and reviewed throughout the project to keep it on track.
All of the problems encountered in Case Study A could have been overcome if due attention had been paid to the business case by application of the proposed software development framework, as is described in the following:

1. The problem of late requirements could never have occurred if a business case for the project had been prepared as the requirements would have been a necessary part of the justification for the project. The need for marketing input would also have been identified in the early stages. It would have been very unlikely that assigning the project manager to collect the customer requirements would have been the most effective means of achieving the business objectives.

2. The zero defect philosophy would have taken a more natural place in the project priorities when the cost of pursuing the philosophy to the very last defect was compared with the cost of performance failure and of delayed delivery.

3. The full documentation of every last process detail is time consuming and unnecessary. However, some processes are critical for the effective development and quality of a product. A business case approach could be used to know when and where to define a process to BS5750 or any other standard. If the problems caused by not defining a process are likely to cost less than defining and recording the process then this is a valid reason for leaving it out of the standard.

4. Arguments over software choice, such as in the Sybase-Oracle debate, or over any project alternatives could have been resolved immediately by comparing the contribution of each alternative towards the business objectives. If one will cost more in training and in programming time because of the developer inexperience, this should be compared with the company cost of utilising more than one vendor's software. The cost in this comparison can be measured in terms of money, effect on time scales, market position, customer satisfaction or the drain on resources or any other business objective. If there are technical differences the costs associated with the advantages and deficiencies of each should be used in the comparison.

5. Part of any business case would have been a risk analysis identifying the risks and the consequence of each event. This analysis should have revealed the...
potential problems of a single, all or nothing delivery. Incremental developments will nearly always be justified on business grounds.

6. A realistic time schedule would have been necessary for a valid business case to be made. This justifies time and resources being spent on making realistic estimates needed for a project. Although there may be pressures for shortened time scales the allocation of unrealistic schedules can only reduce the chances of achieving the business objectives.

7. An advantage of a realistic project schedule is that resources such as staff allocation would have been properly planned. The costs of panic measures such as the allocation of many new team members at the same time would be avoided. Variations in the schedule should also be justified. The advantages of attending the trade show should be set against the delays in delivery caused. Only if the trade show could be shown to better serve the overall business case should it have been allowed to delay the project schedule.

8. Finally, the need for measurable project goals emphasises the need to adequately test that the project has met its objectives. Had the project objectives been clearly stated then it would have been obvious that the four beta sites were clearly inadequate to measure the success in meeting the objectives.

The most significant conclusion from the GEC Power Instrument & Control Case Study is that a company must address all IT and associated business processes in order to improve the overall software delivery success.

4.5 Conclusion
In terms of the thesis objectives, this Chapter shows the results of not using a framework in which all stakeholders involved in the requirements and delivery of a software application have clear roles, responsibilities and guidelines.

The case study shows that by simply applying quality standards, in this case BS5750 and Crosby, does not by itself lead to quality software. In fact, the 'misapplication' of
quality standards can contribute to the overall failure of the project. Whilst the project may meet time and schedule it can still completely fail in terms of user acceptance.

It can be seen from the survey carried out by White and Fortune (2002) that over half of the companies examined were forced to develop their own Project Management methodologies as today's methodologies do not provide the required level of detailed instruction. A further survey by Fitzgerald (1998) indicated that 60% of the survey respondents did not use any form of software development methodology and that only 6% applied methodologies rigorously. Fitzgerald further indicates that benefits would be gained by the adoption of a framework in which processes, tools and techniques could be added and tailored to different types of development.

Quality standards today are aimed at the post-release cost minimisation for the supplier and not at the rapid development of quality software and final customer satisfaction. Therefore, the need for a much-improved framework is very evident.

It is clear that the adoption of either existing quality standards, or, development methodologies by themselves, will not result in significantly improved software product delivery.

This Chapter has shown that to get the best from standards and project management methodologies what is required is a flexible quality framework, which itself provides a set of basic quality best practices, whilst allowing the integration of standard industry methodologies such as PRINCE2.

The framework must allow various methodologies to be integrated in such a way that their guidelines and templates are made visible together with the frameworks own best practices and most importantly the actual projects artefacts.

Therefore from this Chapter it can be concluded that a new quality framework is required which take as its basis the following design principles:

1. The framework must ensure, as a first priority, a clear understanding of the company's overall business objectives and how the software project aligns with these business objectives.

2. The framework must provide a basic set of quality processes for software delivery
3. The framework must provide practical guidance on how to carry out each software quality process in practice.

4. The framework must provide the capability to integrate industry standard methodologies. This will improve the timescale for introducing the framework, both in terms of adopting a new methodology and for those organisations who are already using a specific methodology.

5. The framework must include all levels of quality processes from Programme Management level to Specialist Software Development level.

6. The framework must be able to bridge between management processes, which are largely waterfall in nature and the new iterative development methodologies.


8. The framework must provide a method for measuring the quality improvements as this will be required to justify the cost of introducing the framework and to show to external audit bodies.

9. The framework must provide all framework artefacts to all management and staff who have a stake in the programme or project, in such a way that the artefacts are easily accessible, logically structured and under version control.

10. The framework must address the impact of software development right across the business and consider all aspects from stakeholder involvement to delivery and deployment.

As stated in the Chapter outline earlier the conclusions drawn from the analysis of existing standards and project management methodologies, coupled with a literature review and a detailed case study has generated a set of requirements that will be used in the following chapter to develop a new quality framework.
5 Chapter 5 - Design of the new quality framework

This chapter describes the fundamental design of the new framework, taking as its basis the design principles as established in the previous chapter.

Whilst the chapter describes the core functionality of the new framework, which has been given the name QFSD (Quality Framework for Software Development), the design criteria such as: scalability of the framework, integration of third party methodologies, waterfall vs. iterative methodologies, process performance, measurement of capability and online framework management environment, will be described in detail in subsequent chapters. Therefore, this chapter will concentrate on the core model only and how it was verified by the researcher against a number of software development products.

5.1 Introduction

The QFSD framework was not designed as a complete solution then applied to software projects from day one in its entirety. Rather the core of the framework was defined, following the initial case study at Fisher-Rosemount and then extended by applying it to further projects over a number of years at Fisher-Rosemount. The framework was subsequently extended based on larger and more commercial and business transformational projects at Celesio.

In total, the framework has been iteratively developed over a period of seven years and applied to in excess of 200 software projects by teams ranging from 5 to 25 development staff. These projects range from safety critical software to commercial point-of-sale and web based in-store projects.

In both Fisher-Rosemount and Celesio the result has been a significant improvement in schedule predictability (87% of projects meeting their original schedule dates), 81% of projects coming in on cost with no projects being cancelled. Previously in Fisher-Rosemount only 64% came in on schedule and 35% on budget. In Celesio, the figures were very similar. Note that a small number of projects were cancelled for business reasons, but no cancellations were due to IT timescale slippage, quality or cost overruns.
5.2 The scope of a quality framework

Based on the conclusions as described in Chapter 4, the new framework must cover all aspects of software development from project initiation and definition to final delivery. Many software methodologies only cover specialist parts of the overall software development lifecycle, or attempt to cover an extended set of lifecycle elements, but still focus on a single key element such as project management or software design and implementation. For example: PRINCE2, (The Stationery Office, 2002) or Rational Unified Process (Kruchten, 1999).

5.3 Development of the QFSD Model

5.3.1 Creation of the core QFSD Model

Figure 5.1 shows the complete QFSD framework model based on the ten design principles in Chapter 4. However, this is the result of a progressive definition and refinement of the model over many projects and project types in two separate and very different company cultures.
The initial model was created following the first Fisher-Rosemount and GEC Power Instrumentation and Control case studies as described in Chapter 4.

As Chapters 6 through 10 contribute to the creation and progressive validation of the model, Figure 5.1 will be used at the start of each chapter to show how each chapter contributes to the frameworks completion. This should also enable the reader to immediately see how each chapter fits in to the overall development of the framework.

Whilst subsequent chapters will build an understanding of the QFSD Framework, the following is a brief summary of the elements shown in Figure 5.1.

- **QFSD Repository**: This is the central database, configuration and version management repository of the framework. All documents, source code and third party tool data is managed by this repository.

- **Web User Interface**: This interface enables access to the framework from any supported Web Browser.

- **Org, Mgt, Support & Imp**: These are the 'core' framework software process definitions.

- **Process P'Mance**: This is the process performance tools set.

- **Process Capability**: This is the tool-set for applying process capability monitoring.

- **Tools / Interface**: These are the supporting tools provided with the framework itself.

- **Third Party Tools Interface**: This is the generalised interface to other industry standard methodologies such as PRINCE2 and the Rational Unified Process (RUP).

The initial QFSD model covered the following processes in detail:

- Organisational processes
- Management processes
- Supporting processes
- Implementation processes

Following the first Fisher-Rosemount Case Study, the researcher reviewed the lessons learned from the first project and proposed a fundamental change in the approach to the software development process.

It was clear that the ISO quality standards used in Fisher-Rosemount did not really give sufficient guidance to development teams and project management on the fundamental processes required to deliver quality software (and indeed what constituted quality software). This lead to a change in approach to the definition and inclusion of software processes in the development groups working practices. The new approach was to concentrate initially on defining and improving core software development processes and mapping these on to the ISO 9001-TickIT standards where applicable, rather than starting with the ISO model and bending / adding processes to satisfy the standard. A new set of software development processes were defined and captured in the emerging framework based on experience at Fisher-Rosemount and later, Celesio.

The introduction of the new framework could not occur overnight. Therefore, the approach was to define the framework progressively, with a set of core software processes as the initial starting point. Each set of processes were then proved, refined, and added to, over the period of eight projects prior to being used on the project in the second Fisher-Rosemount case study.

The table 5.1 gives details on the seven projects used as the initial development and testing ground for the new framework.

The table shows improving schedule and cost predictability for projects at Fisher-Rosemount as the framework was progressively introduced and improved on each project.
<table>
<thead>
<tr>
<th>Project Type</th>
<th>No of Lines of Code Changed</th>
<th>Schedule Success</th>
<th>Budget Success</th>
<th>Total Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client / Server Graphical system with relational database (New Development).</td>
<td>23,500 lines.</td>
<td>Planned duration 14 months.</td>
<td>9% Overspend.</td>
<td>96 Man Months.</td>
</tr>
<tr>
<td>Client / Server configuration system with a relational database (Porting).</td>
<td>19,300 lines.</td>
<td>Planned duration 16 months.</td>
<td>On budget.</td>
<td>217.5 Man Months.</td>
</tr>
<tr>
<td>Client / Server configuration system with a relational database (enhancement)</td>
<td>16,700 lines</td>
<td>Planned duration 9 months.</td>
<td>On budget.</td>
<td>144 Man Months.</td>
</tr>
<tr>
<td>Client / Server configuration system with a relational database (enhancement)</td>
<td>12,400 lines</td>
<td>Planned duration 7 months.</td>
<td>6.5% Overspend</td>
<td>120 Man Months</td>
</tr>
<tr>
<td>Client / Server configuration system with a relational database (enhancement)</td>
<td>10,200 lines</td>
<td>Planned duration 7 months.</td>
<td>10% Under spend</td>
<td>75.6 Man Months</td>
</tr>
<tr>
<td>Client / Server Graphical system with relational database (New Development).</td>
<td>16,200 lines</td>
<td>Planned duration 16 months.</td>
<td>10% Overspend.</td>
<td>180 Man Months</td>
</tr>
<tr>
<td>Graphical PC application.</td>
<td>8,300 lines</td>
<td>Planned duration 6 months.</td>
<td>On Budget.</td>
<td>6 Man Months</td>
</tr>
</tbody>
</table>
The new core framework is not a static set of procedures, but rather a dynamic set of processes whereby development processes are reviewed and improved for each project. In fact, process improvements are included as part of each project and are treated like any other feature requirement. This process is later formalised in the framework to provide the process performance and process capability components.

The initial framework model, established following the first case study was based on five major steps as defined in a conference paper by the researcher (O'Neill and Dawson, 1999) illustrated in Figure 5.2.

![Figure 5.2: Five fundamental steps in establishing a quality framework](image)

1. **Step_1** requires that the company has in place a mechanism, which continuously evaluates the company's objectives and disseminates the objectives to the project level. This is perhaps the hardest step to introduce, as it requires open communication from management and can involve a change in a company's culture. This step is introduced to comply with design principle 1 in Chapter 4.

2. **Step_2** is critical at the project initiation and at any point where an alteration or extension to the project scope is considered. This step is also needed to comply with design principle 1 in Chapter 4.
3. Step_3 is the main project implementation where all processes must be carried out with the business objectives in mind. It is important that the processes not only produce the means for working towards the goals (design principles 6 and 7 in Chapter 4) but also allows tracking and auditing to enable progress to be measured (design principle 8).

4. Step_4 is the measurement of the project performance, which must be regularly reviewed, to ensure that continuous improvements and correction to both product and development processes can be made. This complies with design principles 8 and 9 in Chapter 4. At each review, the project performance is reviewed and the necessary adjustments to scope, schedule and resources are made and communicated.

5. Step_5 is a process, which should be applied at the end of a project. The capability of each project processes, such as inspections, defect repairs etc. are reviewed and improvements planned in a subsequent project. This is also required for design principle 9 in Chapter 4.

Each step maps on to the initial QFSD model components in the following way:

- Organisation, Management, Supporting and Implementation component (Steps 1-4)
- Process Performance component (Step 5)
- Process Capability component (Step 5)

In order to ensure that the core QFSD processes were in-line, but expressed in a more practical format, with the major standards (complying with design principle 5 in Chapter 4), they were cross checked by the researcher with the following:

- ISO 9001, 1994 "Quality Systems - Model for quality assurance in design, development, production, installation and servicing.

The comparison involved mapping of the QFSD framework processes to both ISO 9001's main clauses and the CMM key practices. The mapping is admittedly subjective, others may have interpreted ISO 9001 and CMM differently, which is part of these standards shortcomings, but hopefully there is sufficient objectivity to show that the framework covers the majority of the requirements set by these industry standards. The mapping results can be found in Appendix H.

Whilst the new framework is aimed at addressing the shortcomings in the ISO and other standards, it should provide a company with the ability to still comply with those standards if required to do so.

In terms of specific lifecycles the framework can be adapted for waterfall or iterative methodologies or a mixture of both. This will be explained in the Chapter 9, which shows how a waterfall project management methodology can be integrated in to the framework with an iterative development methodology.

The new framework also addressed the fact that written procedures must be kept to a minimum. There should ideally be only two procedures / guideline documents for a project:

- Quality Framework for Software Development (QFSD)
- Project Management Plan (PMP).

The approach was to leverage online tools and document templates, to impose any process 'rules', with only a single Project Management Plan as being the source of all project specific information, techniques and objectives. In fact, all project documentation, in line with design principle 10, should be made available in a single online repository; this includes the actual project artefacts such as requirements and design documents. A first prototype of this new QFSD online repository was used as part of the second Fisher-Rosemount Case Study in order to validate the above assumptions.

The initial QFSD processes were written as a set of generic processes published in the QFSD online repository, from which the specific development group or company processes could be tailored. Note that other procedure documents may be required.
for specific developments. This approach does not preclude them, but suggests that it is necessary to ensure they are required, are flexible and are not just being provided to satisfy a clause in the ISO or other standard that does not really contribute to the quality of the development.

As part of the second case study the initial project management plan (PMP) template was also introduced in order to start to address the project management component in the new framework. More details on the project management components and its associated project management plan are discussed later in the thesis.

The initial QFSD processes address the software production and controlling processes, which includes a basic level of project management. The processes within this document are applied to specific projects by translating the processes that are pertinent to a given project, through the medium of the PMP and any associated software development tools. If a process exists in the QFSD online repository, but does not apply to a specific project, then the exception is documented in the project management plan. Whilst this is simple, it is necessary to ensure that all exceptions to the standard framework are documented, as these will cause a failure during an ISO or other accreditation audit. During the researcher's employment at Fisher-Rosemount the QFSD framework enabled the company to attain the ISO 9001-TickiT accreditation and to retain it over the next four ISO audits.

The initial QFSD model processes (which are aligned with ISO/IEC DTR 15504-7, 1997) are as follows:

1. **Organisation Processes**: These processes are aimed at making sure that software projects are correctly aligned with the business objectives of a company. The company must create the right environment and make its goals clear to all resources that contribute to the definition, development and delivery of software products. In addition, the organisation must be actively involved in the software project to ensure that any changes in company goals or changes in development direction are understood and communicated at all levels. To achieve this the company must have a business aligned IT strategy and associated business IT strategy.
2. **Management Processes:** This section contains key elements which are often under utilized. For example, Monitoring and Control is not a passive cost monitoring process. Monitoring and Control needs to be used as the basis of detailed task understanding for developers and involves estimating, risk management and continuous improvement of the processes themselves. Improvements and initiative should not be stifled by strict adherence to procedures.

3. **Supporting Processes:** These are supporting processes, which run asynchronously from, but in conjunction with, all project execution phases. These processes are required to ensure that a software project is delivered with a high level of quality. An example would be configuration management.

4. **Implementation Processes:** Development phases are the traditional phases within which a software project will progress through (concept, requirements, design, implementation, test, system integration and release to production). This process needs to be adapted to the best possible approach for a given size and type of project.

Each of the four generic processes, groups together a set of detailed sub processes, which form the basis of all tasks within a generic software project and are shown in Table 5.2.

**Table 5.2 Core QFSD Processes Model**

<table>
<thead>
<tr>
<th>Organisational Processes</th>
<th>Management Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company Objectives</td>
<td>Project Management Plan</td>
</tr>
<tr>
<td>Business Case</td>
<td>Project Planning</td>
</tr>
<tr>
<td>Project Objectives</td>
<td>Release Strategy</td>
</tr>
<tr>
<td>Project Approval Request</td>
<td>Resource Strategy</td>
</tr>
<tr>
<td>Requirements Definition</td>
<td>Requirements Capture</td>
</tr>
<tr>
<td>Executive Test Plan Review</td>
<td>Development Process</td>
</tr>
<tr>
<td>Executive Release Readiness</td>
<td>Risk Management</td>
</tr>
<tr>
<td>Process Improvements</td>
<td>Monitoring &amp; Control</td>
</tr>
<tr>
<td></td>
<td>Project Review</td>
</tr>
<tr>
<td>Support Processes</td>
<td>Implementation Processes</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Configuration Management</td>
<td>Approach Definition</td>
</tr>
<tr>
<td>Document Management</td>
<td>Requirements Definition</td>
</tr>
<tr>
<td>Quality Assurance</td>
<td>Software Design</td>
</tr>
<tr>
<td>Verification (inspection)</td>
<td>Software Construction</td>
</tr>
<tr>
<td>Validation</td>
<td>Defect Repair Strategy</td>
</tr>
<tr>
<td>Third Party Software &amp; Reuse</td>
<td>System Integration</td>
</tr>
<tr>
<td>Defect Management</td>
<td>Release Testing</td>
</tr>
<tr>
<td>Process Improvement</td>
<td>Organizational Readiness</td>
</tr>
<tr>
<td>Change Control</td>
<td>Software Maintenance</td>
</tr>
</tbody>
</table>

However, not all contributors to a software project are involved with, or interested in, every process. Therefore, the QFSD and associated project management plan and development environment must be structured in such a way as to enable contributors to access relevant information quickly.

A survey was carried out by the researcher within the Fisher-Rosemount organisation, to determine which processes were relevant to the various contributor groups within a project.

The contributor group consists of twenty key staff divided between senior technology and business management, IT project management, operations & support and a subset of customers. The senior technology contributors were the development managers associated with each of the major systems that together comprised the main control system product range from Fisher-Rosemount. The business management consisted of each product manager for each of the corresponding major systems. The project managers were those project managers with at least 5 years experience of managing large projects with the company. Operations and support contributors were selected on the basis of providing initial deployments and subsequent long term first and second line support for the major systems. The customers were divided in to two categories, which were end-user customers who
run the Fisher-Rosemount and value added resellers who design and configure the systems for specific customer applications.

The group was selected as a representative cross section of staff involved in all major IT deliverables. Each member of the group received a questionnaire containing the QFSD processes and associated descriptions, similar, but in a shortened form to those process descriptions listed in Appendix C.

Each member of the group was requested to highlight the processes that they thought were relevant and the level to which they required information on each. This was done by making the questionnaire available to them in the form of a web page and the information then collected and consolidated.

The response to the questionnaires was high with seventeen fully completed questionnaires out of twenty with all groups represented. Refer to Appendix I, Table I1: QFSD Process contributor relevance questionnaire.

The results were as follows:

<table>
<thead>
<tr>
<th>Process</th>
<th>Interested Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organizational</td>
<td>Senior Management / Project Management</td>
</tr>
<tr>
<td>Management</td>
<td>Project Management</td>
</tr>
<tr>
<td>Supporting</td>
<td>Project Management / Team Lead / Developers</td>
</tr>
<tr>
<td>Execution</td>
<td>Team Lead / Developers</td>
</tr>
</tbody>
</table>

This is what you would expect given the way contributory groups are traditionally partitioned. However, it could be argued that the entries in the interested groups section should be 'All'. This is because if contributors understand each aspect of the project, not necessarily in detail, but to a level that involves appreciation of why things are being done in a certain way and the background behind project decisions, then the project will be more successful.

An example of this wider communication was the introduction of project meeting minutes, issued promptly following each project meeting, a simple but very effective
process. This process was instituted for all projects at Fisher-Rosemount. The result was that all developers and managers were kept up to date with the project progress, problems and decisions.

A significant portion of project post-mortems the researcher has attended in his career has always had the issue of poor team communication as a major issue (Von Zedtwitz, 2002). In fact, one specific project post-mortem resulted in a particular development group claiming that their performance was hindered, as the project manager did not pay enough attention to the group. This attention turned out to be more one of their need for the management to appreciate what they were delivering, rather than any additional practical help / guidance the manager could have given them. Therefore, communications in a project can take many forms and needs to be addressed whenever the issue is raised.

Communications between software teams is the subject of a larger discussion (Martha et al, 2006). The advent of e-mail communication has limited 'face to face' discussions with colleagues, even if they are only several feet away. This has an impact on problem resolution, design quality and implementation efficiency.

These communication issues are one of the major drivers behind the decision to introduce the QFSD Framework using the medium of intranet technology.

5.3.2 Top level description of Core QFSD Processes

Whilst the full core QFSD model processes are described in Appendix C, this section provides an overview of the processes in terms of what each process is and why it is necessary. Note that only the Organisational and Management processes are described in detail, as they are substantially different to processes found in other methodologies.

The support and implementation processes are similar to existing methodologies, or given the integration capabilities of the QFSD framework, can integrate and adopt other methodologies directly. However, there are a number of unique techniques used to carry out some of the support and implementation processes. These unique techniques are described separately in Chapter 6 on Tools & Techniques.
Tables 5.3 and 5.4 give the general description of each of the organisational and management processes respectively. In each table the impact of each process is analysed. This gives a justification for the inclusion of each process and relates the process to the needs identified in the literature review in Chapter 2, section 2.5.

Table 5.3: Organisational process descriptions

<table>
<thead>
<tr>
<th>Organisational Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Company Objectives</strong></td>
</tr>
</tbody>
</table>
| • **General Description:** In order for a software product to be successful, it must be aligned with the overall objectives of the company. This alignment must be in terms of looking forward for growth, customer retention, business area, market need etc. IT must also be inline with the company’s revenue generation showing an acceptable return on investment in a given timeframe. For example, an ROI of between 14 and 18 percentage over a 3 to 5 year period was acceptable in both Fisher-Rosemount and Celesio AG.  

There are of course projects, which do not have a demonstrable ROI, such as enabling network infrastructure, but these do and should have a compelling business case, as the investments are typically large.  

• **Impact:** A company that does not align its software developments with its overall business objectives will waste time, effort and funds in the delivery of software projects that do not directly contribute to the company's current and future profitability. (See section 2.2.7).

<table>
<thead>
<tr>
<th><strong>Business Case</strong></th>
</tr>
</thead>
</table>
| • **General Description:** This is the fundamental step of establishing and validating the business need for the software deliverable.  

The business case must be generated as a joint exercise by the business customer and the IT project management responsible for the deliverables. The IT project management is also responsible for the delivery of the projected benefits in addition to the initial delivery of the software.  

• **Impact:** With out a fully agreed business case a software project cannot claim to be aligned with the needs of the business or the end customer. |
Not having a business case is the highest risk factor in software projects chances of success. (refer to section 2.2.7).

**Project Objectives**

- **General Description:** Following on from the business case and in fact contained in both the business case and the project management plan are the project objectives.

  The project objectives define all major and secondary objectives for the software delivery. They are also checked for alignment with the overall company objectives.

  The objects may also included quality improvements as an objective and often this is a very positive method of carrying out quality improvements that are driven by the project managers and developers themselves.

  Objectives are communicated to all staff involved in the project and are used as part of the measurement of success for the project. (See section 2.2.6).

- **Impact:** Without objectives, a project could drift and deliver a software solution that was not inline with the company objectives or the original objectives of the business customer. (See section 2.2.7).

**Project Approval Request (PAR)**

- **General Description:** The PAR is the official document, which grants the project the right to expend company funds in the creation of software, and the ultimate release of software deliverables.

  It also provides a mechanism to gain approval from all responsible parties for the software project to start expending company funds.

  The PAR document is often used as the basis of the project management plan document.

- **Impact:** Approval from all project sponsors is vital in order that joint responsibility for the project is established at the outset. (See section 2.2.7).

**Requirements Definition**
• **General Description:** This uses a modified version of the classical Quality Function Deployment (QFD) approach to software requirements capture and producing formal software requirements specifications, complete with a unique requirements numbering scheme, to be the most efficient approach. The modified QFD technique is described in Appendix C under requirements definition.

The modifications are based on a considerable number of successful projects delivered by the researcher's development teams in both Fisher-Rosemount and Celesio AG.

The requirements and their prioritisation are carried out in a systematic way by a cross departmental team. This team consists of marketing, operations, support, technology and senior management. By taking this approach, the requirements are prioritised in the best interests of the company and are assigned an appropriate priority level.

Whilst the requirements are crosschecked against the design, the key is to have a method of holding each requirement against its equivalent release test case. This ensures that each requirement can be tested and verified as tested during the final release-testing phase.

A simple method to achieve this is to write a set of test case procedures (Test Procedure Specifications). However, the requirements numbers should not be embedded against the test cases in these procedures, as this will prevent them from being used for regression testing in a later release. Instead, a cross reference tracking spreadsheet or database should be created, in which each test procedure, set of test cases, associated requirements and test results are recorded.

Taking this approach ensures that all requirements are tested and it also enables test statistics to be generated and test cases to be used as regression tests in later releases.

The modified QFD technique is described in Appendix C under requirements definition.

• **Impact:** The correct validation of requirements as being appropriate for a given software release and their tracking through the testing phases are critical steps in the software planning process. Projects that do not carry out this level of requirements verification with customers and stakeholders will never be fully successful (See section 2.2.7). It must be remembered that success of not just the
delivery of quality software, but the delivery of the correct and appropriate level of functionality.

**Executive Test Plan Review**

- **General Description:** Testing is critical and often the first area of a software project to be cut back in order to meet schedule dates (Brooks, 1995). Cutting back on testing is seen as a 'soft target' and is more likely to be accepted by the business sponsor than cutting back on functionality.

  The idea of the executive test plan review is to establish joint responsibility for the testing right at the outset of the project, such that the business sponsors understand what is required and what the impact will be if the planned testing phase is shortened. This joint ownership is also vital in the later project stages when resources and assistance are required from the other groups. (See section 26.1.1.6.1).

- **Impact:** Remember that the resources that may be needed in technical support or manufacturing, have different reporting structures. Therefore the involvement of their senior management in the test plan review helps to give visibility to the project, making it easier to call on resources when required. (See section 2.2.7).

**Executive Release Readiness**

- **General Description:** This meeting is held at the end of release testing and field trials. The objectives of the meeting are as follows:
  
  - To determine if the software product is ready for release
  - To ensure that the organization as a whole is ready for the release and able to provide manufacturing and technical support

  To establish joint responsibility for the release. This includes development, marketing, quality, operations, technical support etc.

- **Impact:** This is a key factor in the successful release of a software product. No matter how good software product is, if the support, distribution and documentation are not available then the release will either be ignored or gain a poor reputation. (See section 2.2.7).
Process Improvements

- **General Description**: A key difference in the approach to quality improvement in terms of QFSD as opposed to other methodologies is the assignment of quality to the project, rather than to a separate quality department or other body.

Real quality improvements are made by the people who carry out the project work and are therefore practical and more easily adopted by their peers.

Whilst the introduction of the QFSD framework will need to be initially introduced jointly, by a company's software development and project management functions. The continued process improvement will be driven by the projects post-mortems and the inclusion of the improvements as part of the upcoming projects themselves.

This philosophy was tried at Fisher-Rosemount and Celesio AG. And was found to avoid the overhead of a separate quality department, improve adoption rates and provide real continuous quality improvement.

- **Impact**: Quality is often assigned to a separate quality department and is driven by the need for the software development and project management groups requirement to pass both internal and external quality audits. Whilst this may inject a certain level of quality or at least traceability into the development process, it does not drive real and innovative process improvements in the development and project management techniques used. This results in the lack of quality process improvement and often an increase in documentation levels in order to satisfy the auditors and to compensate for the lack of real quality processes. (See section 2.2.3).

Table 5.4 Management process descriptions

<table>
<thead>
<tr>
<th>Management Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project Management Plan</strong></td>
</tr>
</tbody>
</table>

- **General Description**: The project management plan document, refer to appendix C for an example, is the key management artefact in any software project.
The document is a living artefact that is initially released at the start of a project and is updated at the completion of each major project milestone. The document provides both project stakeholders and participants in the project with the key project information. For example, the project management plan will describe the framework processes that are being adopted for the specific project, planning approach, stakeholders, objectives, benefits, milestones, responsibilities etc.

- **Impact:** Without such a central document pulling together all organisational, management and project artefacts and objectives, the project would lack the necessary focus and management required in order to make a successful software delivery. (See section 2.2.7).

---

**Project Planning**

- **General Description:** The key to successful software delivery is detailed and realistic planning in order to achieve a real and tangible set of business benefits. This is not to be confused with standard project management techniques. Project planning in this sense is how to size, resource, prioritise and schedule high quality software. Remember that project management techniques are all about planning, but the average project manager does not understand the details on how the software delivery needs to be sequenced. Software planning is a major topic in itself. In order to gain a better understanding, refer to Appendix C 'Project Planning Process' for a detailed introduction.

- **Impact:** The impacts of poor project planning may seem obvious at a first glance e.g. late delivery, additional costs or even project cancellation in extreme cases. (See section 2.2.7). However, the more relevant approach is to be able to spot projects that are starting to go wrong due to poor planning and apply corrective measures. The following are some examples from the experiences at Fisher-Rosemount and Celesio AG observations showing indicators of problems. Note that the list is by no means exhaustive:
  
  - Each project stakeholder has a different view of the projects objectives. This is a clear indicator that the stakeholders have not been correctly engaged in the project planning.
  - The initial project plan's Gantt chart shows an excessive number of
parallel tasks. This is an indicator of a poorly planned project as unless the project runs perfectly and has superb management, it is bound to have slippage with the knock on effect being late delivery.

- All project resources are assigned full-time to the project from day one. This indicates poor resource planning and hence poor overall task/effort planning.

- Milestone delivery dates are missed coupled with the redefinition of both delivery dates and contents. This is a clear indication that the project is going late.

- There is little or no prioritisation of requirements. This shows that the planning has no flexibility in terms of removal of features from a release.

- There is a tendency to cover up late delivery by dividing the deliverables in to a number of phased deliveries. Phased delivery is a good technique, but should be part of the original project planning.

- A revised plan is produced in which the testing phase has been shortened by running all testing phases in parallel. This is a classic indicator of a project going late.

### Release Strategy

- **General Description:** Assuming the software deliverable contains the correct features and is of high quality, it can still be deemed a failure if the release strategy is wrong. In order that a software product to be fully successful it must be fully supported by the company in terms of marketing, operations, support etc. In addition, it must be released to the customer base in such a way that the upgrade process is smooth, well supported with minimum disruption to their ongoing business.

A simple check list might be useful here:

- All parties involved in the manufacture of the delivery materials, sales order processing, artwork, documentation need to be involved in the project and kept up to date throughout.

- The actual materials to be delivered from manufacturing to the customer must be checked and tested to ensure that all materials are
included and that the media is not faulty.

- Marketing must inform customers of the benefits of adopting the new product.
- Technical support must be trained in the new product in good time for the release.
- Any training courses must have been established and beta tested.
- The migration process from all versions of the software must also have been tested and technical support trained in the process and associated issues.
- Care must be taken if delivery media have been changed or if license key technology has been changed. Customers can delay upgrades due to having to learn a new process or having to purchase a new type of media reading device.

- Impact: The impacts of not having a well-planned release strategy are many, but the de-motivation this inflicts on the development team who have delivered a good product is not to be underestimated. (See section 2.2.7). An example can be taken from a recent product section process at Celesio AG, managed by the researcher. The selection process involved a standard supplier selection process in order to select a company standard Enterprise Integration Tool (EAI). The result was that the product with the best technical fit to the requirements of Celesio was not chosen. The reason was that each customer reference site complained that whilst the EAI product was very good, they could not migrate to the later versions as this was always a major step, with little or no support from the supplier. In effect customers were stuck on the version that they first received after taking on the product. In terms of the supplier they were supporting all versions of their software from version 1.0 through to version 5.0.

---

**Resource Strategy**

- **Description:** Availability of suitably qualified resources is always a major issue when planning projects, especially if there are a number of projects competing for the same resources. Obviously, the types of resource required varies depending on the type of project and the extent to which the management of development is outsourced to a third party or indeed parties. Management of third party resources
is difficult and requires close project management co-operation between the project owners and the third party. This is even more challenging when the third party is offshore. However, generally the progressive ramping up of resources on projects followed by a gradual winding down shows a profile that is in line with a well-planned project. This preferred resource profile is shown in the second case study described in Section 26.1.2.4.

- **Impact:** Again the failure to manage a project's resource strategy will cause the project to be unsuccessful. Classic issues are either front end or back end loading of resources on to a project, failure to have the right skills available and failure to check the skills of third parties, especially when the third party is offshore. Whilst offshore development companies tend to have a high level of quality accreditation, other areas need to be checked such as familiarity with the required technology, level of English, time difference etc.

### Requirements Capture

- **Description:** If the requirements definition has been completed using the modified QFD technique, requirements capture is a standard process (e.g. Chatzoglou and Macaulay, 1996)

- **Impact:** Failure to capture the requirements accurately and translate them into a form that can be understood by both the customers and the development team will result in a software delivery that does not meet the original customer expectations. (See section 2.2.7).

### Development Process

- **Description:** An industry standard development methodology such as the unified process (UML) can be used successfully with the QFSD framework, as can other development methodology. However, the methodology developed by the researcher as part of the QFSD framework, and described in the Tools & Techniques chapter, has been named 'Clusterisation'. This methodology has been used to deliver a significant number of projects at both Fisher-Rosemount and Celesio AG.

The approach is incremental rather than purely iterative in nature. Clusterisation enables a software development project to decompose the development into a number of 'vertical slices' of functionality. Each vertical slice is assigned to a
development sub-team and on delivery provides a working system that can be tested or form the input to the final release testing phase.

- **Impact:** The clusterisation approach provides a number of advantages. (See Section 6.5).
  
  - It enables the development and all associated project management activities to be managed in smaller pieces. This makes the project less complex from both the development co-ordination and project management perspectives.
  
  - As each cluster delivery provides an entire slice of working functionality, this can be tested and facilitate early defect fixing before going to final test. Provides a working reference system for other clusters to test their functionality against.
  
  - Each completed cluster could be taken as the final enhancement and allow the entire product to be moved to release testing. Note that in the technique all high priority customer enhancements are included in the initial clusters.
  
  - Each cluster deliverable enables a working test system to be made available for customer requirements validation.
  
  - Project management is structured around each cluster.
  
  - Performance and capability improvements can use the more granular information provided by planning and executing a project on a multiple cluster basis.

---

**Risk Management**

- **Description:** The approach used in the QFSD framework is derived from a number of government risk management standards, as defined in appendix C. The approach includes the following techniques:
  
  - Risk avoidance
  
  - Risk reduction
  
  - Risk retention

Released Version 1.0
- Risk transfer

These techniques are equally applicable to both in-house and third party external software developments. The main philosophy is that risks are actively managed throughout the project, as old risks will be avoided or resolved as new ones are added to the risk management plan. The risk management plan will include risk descriptions, likelihood of occurrence, precautions, contingency plans and severity of the risk in terms of both cost and schedule.

- **Impact:** The consequences of not having a risk management plan is that risks cannot be avoided and responses have to be put in place quickly and are often either inadequate or only partially successful. (See section 2.2.7).

### Monitoring and Control

- **Description:** Focussing the project management on the monitoring of development clusters effectively breaks a large and complex project into a series of smaller, more easily managed sub-projects. The technical used is to plan, execute, monitor performance, capability, functionality, level of defects, costs and schedule on a cluster-by-cluster basis. This information is then used to adjust subsequent clusters in order to continuously optimise the projects overall performance (O'Neill and Dawson, 2005).

- **Impact:** If a project is not monitored on a frequent basis it will quickly go out of control and that will inevitably lead to its failure. (See section 2.2.7).

### Project Review

- **Description:** This is a structured review process, which should be defined in the project management plan and used to keep the project on track, monitor quality and enable early corrective action to be taken. Given that the clusterisation approach is used the detailed points for review can be found in Appendix C.

- **Impact:** Not carrying out structured project reviews will lead to the project drifting with schedule and quality issues going unchecked. (See section 2.2.7).

- **Description:** This process must ensure that all requirements are fully tested. A simple rule is that all requirements must carry a unique number and they must be held against the test case that is being used to verify that the requirement has
been translated into the software product. The requirement to test case mapping is best done using a database, in which the requirements, test cases and resulting test logs can be recorded. The test phases applied are standard (Knott and Dawson, 1999) and can be seen in Appendix C.

- **Impact:** Apart from inadequate testing, not tracking the requirements against test cases, which are often a many-to-many relationship, can mean that requirements are not tested or worst case never make it into the software product. (See Section 2.2.7).

**Problem Resolution**

- **Description:** Problem resolution is a set of ground rules that are put in place at the start of a project and are designed to minimise the impact of problems that are outside of the project's control. Examples of such problems are investment levels could be reduced part way through a project, or a supplier of a software component goes out of business with no prior warning.
- **Impact:** Not having problem management technique in place can lead to delays caused by inactivity in the project. (See Section 2.2.7).

### 5.4 Support of design principles

Whilst the core QFSD process model addresses a number of the QFSD design principles as listed in the previous chapter, there are a number of supporting tools and techniques that complete the design coverage.

Table 5.5 describes how the ten design principles derived in Chapter 4 were incorporated into the overall QFSD framework.

<table>
<thead>
<tr>
<th>Design principle</th>
<th>Addressed in Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The framework must ensure, as a first priority, a clear understanding of the company’s overall business objectives and how the software project aligns with these</td>
<td>The core QFSD process model takes as its first two critical and mandatory processes: Company Objectives and Business Case. These are critical in every software project to ensure that</td>
</tr>
</tbody>
</table>
2. The framework must provide a basic set of quality processes for software delivery. The QFDS core processes, as listed in Table 5.2 and described in Appendix C, provide the basic set of quality processes to enable a software project to deliver successful solutions.

3. The framework must provide practical guidance on how to carry out each software quality process in practice. The core QFSD processes are described in Appendix C and follow the structure:
   - Process description
   - Objectives of the process
   - Techniques used in the process
   - Process tools (if applicable)
   The guideline is available as both documentation and as associated online HTML pages.

4. The framework needs to have the scalability to handle small software projects up to large multiple, interrelated projects within an overall business change programme. Scalability is addressed in Chapter 10. However, the elements of project and programme management were added to enable the framework to scale for single large projects and programmes containing multiple projects.

5. The framework must provide the capability to integrate industry standard methodologies. This will improve the timescale for introducing the framework, both in terms of adopting a new methodology and for those organisations who are already using a specific methodology. The framework has been integrated with two industry standard methodologies. The first is a project management methodology called Prince 2. The second is the iterative development methodology Rational Unified Process (RUP).

6. The framework must include all levels of quality processes from Programme Management level to Specialist Software Development level. Table 5.2 shows that the framework contains a full set of processes that cover organisational project management, support and implementation processes.
7. The framework must be able to bridge between management processes, which are largely waterfall in nature and the new iterative development methodologies. This is best shown in Chapter 10 which shows how the largely waterfall project management methodology is integrated into the framework with the highly iterative development methodology RUP.

8. The framework must provide a method for measuring performance of the quality processes. This is achieved by the inclusion of the process performance part of the framework.

9. The framework must provide a method for measuring the quality improvements as this will be required to justify the cost of introducing the framework and to show to external audit bodies. This is achieved by the inclusion of the process capability part of the framework and is shown to be successful by enabling Fisher-Rosemount to achieve and maintain its BSI TickIT accreditation.

10. The framework must provide all framework artefacts to all management and staff who have a stake in the programme or project, in such a way that the artefacts are easily accessible, logically structured and under version control. The artefacts must be accessible from any managers or staff member's desktop, development workstation or remote access device. This is achieved by the introduction of the online, web based, QFSD repository. This is further described in the Chapter 7 and Appendix M.

5.5 Conclusion

This chapter has concentrated on the overall design of the QFSD framework. The complete framework was designed and validated over a period of seven years and applied to many different types of software project. In this chapter the design of the initial 'core' QFSD framework is detailed, which covers the basic Organisational, Management, Execution and Support processes. In addition, the chapter indicates how the 'core' framework was validated against real projects in Fisher-Rosemount. In fact, the approach had also been applied to warehouse refit projects, which covered building works, automats, networks, hardware and software. A marked improvement was also observed in application of the framework to these mostly non IT projects.
The chapter also indicates how the design of the 'core' QFSD framework maps on to the original design principles as defined in Chapter 4.

Subsequent chapters will introduce the extensions that have been made to the framework in order to address process performance, process capability, project and programme management together with the tools and techniques developed within the overall framework, providing both a detailed description and how they were validated. Chapter 7 contains the second Fisher-Rosemount Case Study, which provides a very detailed example of how the 'core' QFSD model and prototype online repository were applied to a major new software product build.

The full list of 'core' QFSD processes, along with overview descriptions can be found in Appendix C.
6 Chapter 6 - Tools & Techniques

This chapter describes the tools and techniques that have been applied to projects that have used the QFSD framework. The techniques presented in this section are those that are sufficiently different to those used in other frameworks or methodologies. The tools described are those that are essential in order to support the techniques as described. For an exhaustive list of techniques used in the QFSD framework, please refer to Appendix C.

Figure 6.1 shows the elements of the framework that are specifically validated in this and previous chapters. Note that the areas addressed are indicated in the figure using bold, underline and italic text. The basic core framework has been validated in the previous chapter including: Organisational Processes, Management Processes, Support Processes, Development Processes and a rudimentary repository.

![QFSD framework validation and progress chart - tools & techniques](image)

Figure 6.1: QFSD framework validation and progress chart - tools & techniques
6.1 Introduction

The QFSD framework consists of processes, techniques and tools, which are tailored to specific projects depending on their complexity, scope and quality level targets. This tailoring must be recorded in the associated Project Management Plan to justify their exclusion. The detailed QFSD framework processes are detailed in Appendix C, along with guidance on the associated techniques and tools. However, in this chapter the tools and techniques that are considered unique to the framework or modifications of existing techniques are developed further.

The tools that are included in the chapter are as follows:

- The online QFSD electronic management repository
- Process performance statistical software package and capability database
- Inspection utility

The techniques that will be included in the chapter are as follows:

- The 'Clusterisation' technique
- QFD software requirements capture and prioritisation technique
- Defect rate quality and release indication technique

6.2 Online QFSD electronic management repository tool

The objective of this section is to describe the QFSD online repository and to show how it supports the QFSD framework. In addition, the section will demonstrate that the repository meets the ninth design principle for the QFSD framework as described in Chapter 3. This principle states "Framework must provide all framework artefacts to all management and staff who have a stake in the programme or project, in such a way that the artefacts are easily accessible, logically structured and under version control. The artefacts must be accessible from any managers or staff member's desktop, development workstation or remote access device."
The initial prototype repository was introduced in Fisher-Rosemount and initially used on projects such as the Control Desktop P2.0 (Fisher.Rosemount Case Study Three as described in Chapter 7.

In line with the fourth design principle, a major update to the repository was made for its introduction in Celesio AG. This update enabled the repository to be scaleable and provide an open interface concept for integration with other methodologies such as the Prince2 project management methodology and the rational unified process (RUP) development methodology.

As stated in the fifth design principle for the QFSD framework, the online repository must be able to integrate with industry standard methodologies. This level of integration flexibility, both in the framework and the online repository is necessary in order to take account of pre-existing methodologies, or to enable industry best practices to be adapted in to the framework.

In terms of validation of the revised repository, it can be stated that the QSFD online repository has been the main software development repository in the UK division of Celesio AG since 1999 and, despite a number of market searches, a suitable replacement has not been found. The search was initiated, as the configuration management application it is based on ‘SourceSafe’ is no longer a fully active product within Microsoft. However, the decision is to move the repository on to another commercially available configuration management tool, rather than move to a commercially available product. Commercially available products tend to focus on either document management or source-code version management. The capability required is document management combined with user configurable storage and access hierarchies, in addition to both configuration and version control.

Given the repository has been in place for six years at Celesio AG, with no user requests for its replacement, this would suggest that it is currently meeting the needs of the company, it is robust and the fact that all projects use it indicates that it is an integral part of the company’s software development process. The online repository is used as the single backbone repository for all software project artefacts. However, whilst the QFSD repository provides source code management, a separate source
code management tool can be used if preferred. Commercial products are just starting to provide single repository support in the area of Service Orientation. This will be discussed in Chapter 11.

The online repository is used as the single backbone repository for all software project artefacts. However, whilst the QFSD repository provides source code management, a separate source code management tool can be used if preferred.

6.2.1 Repository overview

It is assumed that the reader has a basic understanding of the Prince2 project management and Unified Process development methodologies, gained either by prior experience or by reading Chapter 10. In addition, the reader should be familiar with the structure of the product development procedure (PDP), as shown in Table 9.2.

The QFSD framework provides a practical framework from which high quality software products can be delivered. However, in order to move the QFSD framework from a set of documented processes in to a dynamic online management system, the QFSD framework is deployed in the form of an online, web-based management tool, which is made available to all project stakeholders. The following is a list of major stakeholders considered in the design, which is included in the overview in order to understand the type of user interfaces required:

- Senior management
- Marketing
- Project management
- Architects
- Technical leads
- Developers
- Technical writers
- Software manufacturing
- Software technical support
Some of the advantages of the QFSD framework are its adaptability to industry standard methodologies, project types and its flexibility in terms of keeping the processes and tools within it up to date. This process of keeping any set of processes current, presents a challenge with respect to the environment in which the framework is deployed.

Three possible approaches for the delivery of the QFSD framework were considered.

1. The first option was to deploy the framework as a set of process descriptions and check list documents. This approach had been tried in the past within Fisher-Rosemount (known as the SQA initiative), and enjoyed reasonable success. However, on the negative side, it resulted in each project manager implementing a project management and QA structure that was significantly different from the next, which made ISO accreditation impossible at that time. This has the further major disadvantage of non-standardisation in the project management approach used. A knock on effect of this is that a knowledge base of best practices in applying techniques within projects cannot be collected in a meaningful way as techniques are different or applied in a different way on each project.

Another issue is ensuring that all project managers are working to the same document revisions. This would seem to be fundamental, but without a centrally controlled source of procedures and documents the researcher has observed project teams using different versions of standards, and even more frequently, different version of document artefacts.

2. The second approach considered was to purchase a commercial integrated software environment, such as Software Thru Pictures (http://www.aonix.com/stp.html) or Rational Rose™ (http://www-01.ibm.com/software/rational/) which would be customised to support the QFSD model and allow it to be updated with new processes, and modified to record application experiences. This approach was favoured in terms of providing an efficient and tailored application. However, it does have some drawbacks, for example the application would be expensive to purchase, customise and maintain which, in itself, and presents a problem as obtaining funding for project support
tools in any company is very difficult. Generally, products from the market support their own methodology and are not able to meet the QFSD design principle of integrating with other industry standard methodologies.

3. The third approach, and final approach taken, was to take advantage of web technology. Using the basic web technology available from Microsoft Explorer coupled with Java enables the online repository to be deployed on every stakeholder desktop or workstation.

In order to accelerate development the on-line repository can use commercially available configuration management software components. In the current release of the online repository the Microsoft SourceSafe configuration management components are interfaced with the repository. However, other market available configuration management products can and will be used going forward. The reason why a commercial product is not used for the configuration management element is that writing such a configuration application would have been a major undertaking. In addition, it is assumed that development groups will already be using a configuration management product for code management and will require the ability to easily interface with the repository.

Web technology Intel hardware together with the QFSD online site 'published' templates and tools, is all that is required to deploy the QFSD framework. A very basic framework consists of all QFSD processes as web pages, process descriptions, process techniques and a selection of supporting custom or third party commercial tools on top of the repository and it's API (Application Programming Interface). However, it is assumed that the development group or company has established a basic quality framework in to which the QFSD processes can be integrated.

In addition, if the organisation has also invested in other methodologies and wishes to use the QFSD framework in order to integrate them in such away that they have a consistent and complete quality framework, then there is some customisation that needs to done in order to rollout the QFSD framework and online repository.

Examples of typical customisation work are described in Chapter 9.
6.2.2 Repository key features

The QFSD online repository in production today provides the following key capabilities:

- Online web-based versions of the QFSD core processes and techniques as detailed in Appendix C
- Online process performance measurement templates and reporting in the form of graphical charts and reports.
- Online process capability levels database, containing the work products and assigned capability levels for each work product. Refer to Section 5.3 for details of process capability.
- Document and spreadsheet templates for major documents in which the processes are executed
- Inspection utility for each project's artefacts as described in Section 6.4
- User defined product design artefact hierarchical structure and actual artefacts
- User defined procedures structure and actual documents
- User defined project artefact hierarchical structures and actual artefacts
- Interfaces for other shared database tools such as defect tracking and test monitoring tools
- Team general working areas, which are areas on the repository that are dedicated to separate development teams in which work-in-progress documentation is located
- User configurable hierarchical navigation structure (tree browser) which is based on Microsoft's Windows Explorer product, enabling folder based hierarchy generation. The interface is single document in design, using a split window between navigation and artefact icon display
- A facility to open multiple artefacts can be opened in parallel.
• Check-in & check-out, plus historical change history for all artefacts within the repository

The design of the online repository was based on the observation that project tasks, artefacts and reporting are best supported by an integrated online tool-set (Tuman and McMakin, 1997). In the case of Fisher-Rosemount the project management and development teams had a history of using Microsoft desktop tools, which included Microsoft Word®, Microsoft Excel® and Microsoft File Manager®, together with holding documents in the source code repository CMS (code management system).

However, the result was poor communication of project artefacts, reporting, change control, defect history etc. Post release code changes were difficult as the artefacts on which a given release was made could not be collected together as it was impossible to baseline the release and all the related artefacts. The other major implication of this was that a project could not be subjected to an external audit.

Therefore the online repository was designed as a central online repository with basic management capabilities such as version control, check-in / check-out, change history, multiple document formats, baseline and with the ability to integrate with other project and development tools.

6.2.3 Technical design

The repository application consists of a custom web client that runs with either Microsoft Internet Explorer or Firefox. The client Java application is load on to the client machine when the user logs on and makes a connection to the server-based repository. The repository is based on the Microsoft Sourcesafe Database and is accessed by the client API through a COM layer on the server. Connection management to the server is managed by the Microsoft Web Server product IIS.

Clients can access the repository over local LAN, WAN or over the internet. Authentication is provided using either a company’s LDAP directory service, or Microsoft’s Active Directory AD® or Active Directory Application Management ADAM® products.

Each internet page within the repository application uses a common tree browser, which is displayed in the left hand screen frame. Navigation and search buttons are
located in the top right hand frame. Selected artefact pages are either displayed in the lower right hand frame or can be opened in separate windows. The design of the tree browser and display capability was based on the standard Microsoft desktop applications in order that it was familiar to the users and required minimal training. The search function enables full text search on document names and contents. It also provides a summary and potential star rating against the resulting search retrieved listing. Again this was based on common search engines that the users would be familiar with. The tree browser enables the user to navigate through the web pages that comprise the repository application. The tree browser hierarchy content is generated on the fly from a set of user configured data files. Again this was to present the repository in a format familiar to users as it resembled the Microsoft File Manager®, or later Microsoft Explorer®.

All artefacts are stored on the server in a common repository. The structure of the browser and database are determined by the users in order to suit their quality process hierarchy and project development folder structure. Access to specific areas in the tool is controlled by granting a set of rights and restrictions to four types of user role:

1. Super user who has all rights including the right to create the SourceSafe® database
2. Administrator who has the rights to create, delete and modify user profiles and backup / restore the database
3. Read and write user who can add, delete and modify artefacts in the database
4. Read only user who can only read information from the repository

All modifications to the repository are recorded by the user in an audit trail log. The audit trail records all changes to the database structure and content. This enables all changes to be rolled back if required and the database to be recovered. However, only the administrator and super users can carry out a rollback process.
6.2.4 Repository deployment

In order to fully understand how the online repository underpins the QFSD framework the following describes how the repository was configured for use by the Celesio UK software department.

The Celesio UK software department consists of 240 staff, 90 of which are developers, 20 are project managers, and 5 are business analysts. The remainder are operations, support and finance. Therefore, the number of users requiring access to the QFSD repository is large. The Celesio UK software department adopted the QFSD framework during 1999. However, in addition to the adoption of the framework the researcher, in the capacity of IT Director decided that Celesio would adopt the industry standard project management methodology, Prince2. The extension of the QFSD framework to include Prince2 is described in Chapter 10. However, briefly, the Prince2 methodology was customised for introduction in to Celesio and defined, together with the core QFSD processes in a quality document known as the Product Development Procedure (PDP) (Matthew, 2000).

Whilst the repository does provide an extensive range of capabilities, it was decided to keep the initial introduction simple by concentrating on the following benefits:

- Online interactive procedures and processes
- Structured project documentation management
- Structured software product design management
- Team working areas
- Shared databases and tools (specifically the rational unified process tool set).

The following diagram shows the basic architecture of the QFSD repository as introduced in to Celesio UK:
Figure 6.2: Celesio QFSD repository architecture

The diagram shows three storage areas:

1. Sourcesafe, which provides the document storage and management functionality

2. The share, which is a common area used by development teams for the management of artefacts, prior to their formal publication. It is also used as a point to link to the databases in other management products.

3. The repository, which is the database storage area for all artefacts.

The storage areas are integrated in terms of having a common tree browser and cross database search engine. The storage area structures are created by the user and in the case of Celesio consisted of the following:

- Procedures / Processes
- Departments
- Programmes / Projects
- Products
- Teams / Resources
- QFSD administration
• QFSD user guides

All Prince2 artefacts, development artefacts, in this case Rational Unified Process and QFSD artefacts, including guidelines, have a common location for their basic templates. These are used as the basis of all artefact creation and are stored in the repository.

The templates make use of the standard (version, researcher etc) and custom (status, project etc) document properties provided by all Microsoft document types. These properties are used in the revision and release control of the documents. They also form the basis of the online review and approval wizard.

The approval wizard, in conjunction with the document attributes are used to drive the document lifecycle:

• In progress
• Draft
• Ready for approval
• Approved
• Withdrawn

The PDP procedure, which is the Celesio UK software production procedure standards document, defines how the templates are used in terms of processes and the stages involved in the development of a software project.

6.2.5 Repository customer endorsements

The following endorsement summaries are based on feedback from the Celesio UK software department in terms of the benefits provided by the QFSD online repository. The feedback was gathered 18 months after the introduction of the repository by the project management department. A questionnaire was distributed to all staff within the following departments who had applied the framework to software projects:

• Development department (40 members of staff)
• Software engineering department (5 members of staff)
• Project management department (6 members of staff)
• Business analysis department (4 members of staff)
The questionnaire was sent to each member of staff for completion in the form of an online survey. Out of a total of 55 questionnaires 47 were completed and returned, this represents an 85% completion rate.

The questionnaire was designed to be as simple and quick to complete in order to gain maximum number of responses. The questionnaire can be referenced in Appendix J.

The questions were as follows:

• What is your job role?
• Provide a brief description of job responsibilities
• Number of projects in which online QFSD framework was applied?
• What benefits did you observe from the application of the framework?
• What benefits did the framework provide in terms of your job role?
• How could the framework be improved?

The questionnaire results were qualitative, as a quantitative questionnaire would have required a detailed set of questions, which would have been very time consuming for the recipients to complete and reduced the completion rate significantly. However, this questionnaire was designed to get feedback on the actual extent of use of the framework and how staff received the new approach. As can be seen from the summary responses below, the questionnaire did show that the online repository was accepted as a major improvement.

Summarised questionnaire responses:

• Provides a rapid deployment of a quality framework that addresses all areas of the software development lifecycle
• Enables the integration of the Prince2 project management methodology supporting and enabling project managers to deploy and control software development processes in a consistent way across all projects

• Provides the software development processes with a description of their relevance and objectives. This enables the project staff to understand why a process is applicable to a particular project, and how the process has developed to its current level

• Provides a single environment from which the project can be managed and all project artefacts controlled and through which the project can be audited

• Provides an easy mechanism for recording experiences in the application of software process and hence enables the gathering of information for subsequent improvements in the processes and their application

• Provides a supporting set of metrics to monitor the effectiveness of selected processes and their capability maturity

• Suggested improvements included: quality improvement and monitoring tool set, direct interfaces with project scheduling tools and EXCEL. All of these improvements were added to the framework.

6.3 Process performance statistical software package and capability database tools

Using the initial QFSD framework is not sufficient to drive continuous software quality improvement, each process and work product must be evolved and improved in terms of both project team performance and process efficiency. The approach taken in the QFSD framework was to treat both process performance and capability as measurements against which statistical analysis could be used to monitor both the application of a process and the corresponding results when applied to software projects.

Statistical analysis of processes has been used successfully in manufacturing to ensure that processes are repeatable, that the processes are continuously improved
and that performance is improved giving increased confidence in products being delivered on schedule. With this in mind, the application of statistical process assessment enables more rigour to be applied to both the application of processes and the delivery of quality software products.

6.3.1 Overview of process performance and capability database

6.3.1.1 Process performance

A number of process assessment methodologies have been used by the researcher in a number of companies: GEC Measurements, CEGLEC and Fisher-Rosemount. These quality measurement methodologies include:

- ISO 9000-3 TickiT (TickiT January 2001)
- Capability maturity model (SEI CMM)
- Capability maturity model (SEI CMMI)

Whilst these are process assessment methodologies, with a level of capability measurement, they are not systematic and are weak in the area of specific matrix measurements. For example, in CMM a company’s progress through the various CMM levels is determined by an external auditor’s evaluation of the existence of predetermined artefacts, but this does not provide a continuous feedback mechanism during each project.

What the QFSD framework provides is a process-by-process capability improvement measurement, which is applied to every project and hence improvements can be monitored constantly. This constant monitoring enables corrections to be made on a continuous basis and hence increases the rate of capability improvement.

ISO 9000-3 and CMM have weaknesses in that they do not provide sufficient information to the project team in terms of ‘how to’ improve software quality, but rather focus on the audit perspective of identifying artefacts that do not meet the particular assessment methodologies criteria. The assessment methodologies are also very slow to evolve in meeting the needs of evolving project management and development techniques. Therefore, a pragmatic and highly flexible performance and
capability methodology was created as an integral component of the QFSD framework.

The performance methodology involves the use of statistical analysis of project information provided at both the start and end of each major project milestone, or indeed across many projects, which highlights processes that are out of preset control limits and hence need to be addressed. In terms of the development, this includes every major artefact delivery within a project.

To facilitate the analysis and generation of statistic reports a commercially available software package was used ‘SQCpack™’. The statistical methodology chosen was a combination of control diagrams and standard deviation charts. These methodologies enable the results to be shown graphically, which is a major aid in focussing the project teams in areas that require improvement. Also, by using control charts, the level of performance target can be continuously raised by narrowing the acceptable control limits.

The types of information required are deliberately kept to those that are generated naturally as part of either the project management or the project development processes. Special information gathering is not required and hence, the performance measurement process is integrated with the standard end of milestone review and is therefore non intrusive.

6.3.1.2 Process capability

Process capability monitoring is fundamental to continuous process improvement and is partly fed by the performance metrics described in the previous section.

In order to provide a continuous process capability methodology, the QFSD capability approach focuses on a defined set of work products that have predefined inputs and outputs for each QFSD process. Each work product is given a capability level between 0 – 3 and measured first by its existence as a delivered product in a given project and second by its quality. The quality achieved is measured in terms of a percentage 0% – 100%, where 0% is an artefact that does not meet the predefined content elements and 100% where the artefact contains and meets all the predefined
content elements. The capability measurements are entered into a database application, such that the capability analysis can be carried out automatically and reports generated for further analysis and capability improvement. This monitoring of capability is applied to each project milestone and hence provides a closed loop mechanism for continuous improvement of both the process and its application within a given project.

Given that both the QFSD and actual project delivered artefacts are located in the QFSD repository, a custom data analysis application is provided within the framework, which ensures that all project statistical information is kept in a single location.

6.3.2 Key features of the process performance methodology

In order to record, store, analyse and report on performance metrics gathered from each software project a third party tool was selected. Whilst part of this type of tool could have been custom written as part of the QFSD repository, the types of analysis algorithms used needs to be flexible and hence a tool providing a number of options was preferred.

The statistical analysis tool provides an extensive coverage of analysis algorithms, which can be applied to different types of data. For example:

- Control charts for variables, which are used when the data is either based on discrete measurements or based on sample measurements
- Control charts for attributes, which are used when the sample size is itself variable. Attributes are counts rather than measurements
- Custom control charts, which contain both variables and attributes, used when monitoring multiple software development processes.
- Standard deviation histogram, which is used to show conformance of multiple processes to a predefined sigma level

The 'SQCpack™' analysis product provides the capability to produce the various chart types in graphical report format. Each set of analysis source data is held as a
separate entity within the analysis tool with a copy placed against the specific product or project in the QFSD repository and hence is under version and configuration management control.

6.3.3 Technical design of the process performance methodology

The ‘SQCpack™’ analysis product stores each analysis project in a Microsoft™ SQL database. This enables each analysis project to be synchronised with the main QFSD repository and appear as a navigation icon at the same level as the actual development project to which it is associated.

Process data to be analysed are generated within the QFSD framework and imported using a number of file types into the identified analysis project. Results from the analysis are generated in terms of graphical reports. Both the source data from the ‘SQCpack™’ database and the generated reports are 'checked-in' to the QFSD repository. When the specific project is archived within the QFSD repository the analysis source data and reports are included in the archive process.

6.3.4 Deployment of the process performance methodology

The control diagram approach enables any variables or attributes to be analysed in order to discover significant variances outside the predicted norms, which can then be highlighted and acted upon. The majority of processes within the QFSD framework can be analysed using one of the specific control chart types provided.

In the framework it is usual practice to start the statistic analysis using processes that can be easily measured and to apply the analysis on a frequent basis. It is recommended that a selected set of process statistics be run against each set of milestone deliverables in a given project and again for the whole project. For example, apply the selected process to each milestone and then combine the results in order to get an overall view of process performance across the entire project.

6.3.5 Key features of the process capability methodology

Each process in the QFSD framework has assigned to it a number of input and output work products. These work products describe the characteristics and
deliverables which indicate that a process is being applied correctly. The number of deliverables and their completeness is used to determine the measure of capability of each process. However, not all processes carry the same weight as they represent a progressive improvement in process application sophistication. To address this spread of sophistication, or capability, the work products are further grouped into 3 classes depending on the sophistication of each work product. Table 8.3 provides an example of typical process work products.

In order to generate the initial classifications, an analysis of past projects at Fisher-Rosemount was carried out over a six year period, in order to establish a basic core set (default set) of processes that are always applied and those that emerged as the teams became more experienced. The processes and their classifications were reviewed with the project teams before being finally accepted as part of the capability methodology. It should also be noted that classifications can be changed and new work products added. This is done as part of the normal framework process for process improvement and is generally applied as part of each project post-mortem.

The actual capability measurement can be taken on a project milestone basis, over all milestones in a given project, or across a number of projects. Each work product is assigned a capability level, which is fully achieved when all work products are delivered. A standard work product set is provided in the capability tool for each process in the QFSD framework. For each process the capability level can then be assessed. The capability is measured for each process as it is implemented for each individual project. As the work products are delivered the capability level of the process is increased. The capability level of each process on completion of the project can then be used to determine overall capability of the processes employed by the development team. This is then used to determine which processes need further improvement and development.

It was observed at Fisher-Rosemount that as the development team’s capability improves, then a number of work products start to be delivered consistently and at a very high level of quality. At that point, these work products can be given a lower
capability level and new work products added at a higher level. In this way the overall capability can be continuously improved and the development teams challenged.

6.3.6 Technical design of the process capability methodology

The process capability database consists of a Microsoft™ SQL database with a custom written Visual Basic™ user interface and a graphical report generator. Both the source database and reports can be stored and accessed via the repositories navigator from the QSFD repository. In this way all project artefacts and capability analysis information can be kept in a single location under configuration management control.

The user interface provides the following facilities:

- **Administration**: the administration screens enable the process and associated work products to be created and assigned the required capability levels. They also provide the means to assign the maximum achievable % contribution of each work product to the overall capability target for each process. The administration screen also enables the project to be identified uniquely and for selected processes to be excluded from any given project, which is a key capability from an audit point of view.

- **Process Capability Capture Screens**: there are two types of screen for the assignment of capability measurements. Both enable each project process to be reviewed and the actual measurements to be added. However, with the first type of screen, the measurements are added process-by-process with an associated graphical chart. The second type of screen enables rapid assignment of measurements by presenting all processes that have been assigned in a simple data entry table format.

- **Reports Screens**: the reports can be generated as either onscreen, PDF or HTML. The report types are described as follows:
  - **Process Capability Reports**: this report prints out all or selected processes together with work products, grouped by inputs or outputs,
work product descriptions, the maximum % capability each work product can achieve and the actual % capability achieved.

- **Aggregated Capability Reports:** these reports can be run across projects, within a single project or within a single project's milestone. Processes can be grouped to show % work products that achieved each of the capability levels, reported over multiple projects, single projects, milestones or groups of milestones.

### 6.3.7 Deployment (capability)

The preferred way to deploy the process capability database is to create a single instance of the database and link it into the QFSD repository in order that the projects and individual process capability tools and metrics use the same database. As stated earlier, the database already contains each process and each work product, together with each work product's default capability level (0 to 3). It is assumed that the QFSD processes are applied to the development and that at the end of each milestone a capability measurement activity is carried out.

At the end of the project the aggregation reports will be run to determine overall capability and to assess if any corrective actions taken at the end of a given milestone have had a positive effect on subsequent milestones. Further reports can be run to compare capability performance of the current project with past projects in order to assess both the processes within the framework and the trend towards actual improvement.

### 6.4 Inspection utility tool

The inspection utility was created in order to provide a centralised project inspection tool that enabled project team members to easily execute formal inspections. This is necessary in order to meet the requirements of the QFSD Verification and Inspection Process.
6.4.1 Introduction to inspections

Inspections are different to reviews or walkthroughs in that they are executed in a very precise, controlled and formal way. The classical method for carrying out formal inspections is that proposed by Fagan (1986).

The Fagan method has been applied to all projects that have been used to validate the QFSD framework and has been successful. The following provides a list of the benefits experienced in the many projects that the researcher has been associated with and are further supported by Doolan (1992) who observed the same benefits when applying the Fagan method to projects at Shell Oil:

- The quality of an inspection is much higher than that obtained by a simple review meeting in that the collective comments and discussion that is generated by all participants having formally reviewed the materials and being in a position to discuss comments in open forum at the inspection, results in a major improvement in the overall quality of those materials inspected.

Comparisons of defect rate removal in projects executed by the researcher in Fisher-Rosemount shows a defect discovery rate increase of 35% at the requirements stage. Defects discovered and corrected at the requirements stage are far less expensive to correct than if they make it all the way through to the code testing phase.

- From a cost perspective the earlier in the software lifecycle a defect can be identified and corrected the less time, effort and cost has to be applied in its correction (Fagan, 1986). The Fagan method can and has been applied at all stages in the lifecycle: Requirements, Design, Code, Test Planning, Testing Results and User Documentation. By applying the Fagan method defect discovery is brought forward and hence defects are repaired at lower cost. Doolan (1992) observes that the correction of each defect discovered in the testing phase at Shell Oil uses thirty times the effort used to run the inspection on the associated requirements that would have found and corrected the defect.
• Carrying out these detailed inspections also acts as a way of sharing knowledge between team members which helps protect the project if a member of the team is removed.

• Inspections are a way of introducing new members of staff to the project team’s quality culture.

• For developers having detailed code inspections with peers and more experienced staff is a great way of learning and improving code quality.

6.4.2 Key features

It would be simple to say that the inspection utility provides an automated environment for carrying out Fagan inspections. However, this would assume that the reader fully understands the methodology, therefore the following is a short list of the key features required to support the methodology.

• It should be understood that there are a number of roles in the methodology the most important of which are the producer, moderator and recorder. The producer is the person who requests the inspection and provides the materials. The moderator manages the whole inspection process and the recorder ensures that all materials, comments and minutes are created and distributed to the inspection attendees.

• The producer creates the inspection using the inspection tool, selects the attendees and assigns the roles as described above. Electronic references to the inspection materials are provided and the inspection tool assigns a unique ID to the inspection, creates the necessary working spaces in the tool and informs the reviewers of the inspection date, venue and materials to be inspected.

• Review comments are entered into the inspection tool by the reviewers and assigned a severity level: major, minor, related issue, good point, question and a classification: clarify, incomplete, inconsistent etc.
• Prior to the meeting the producer reviews the comments, addresses duplicates, typos and the like and then provides the reviews with a complete list of comments for the inspection meeting.

• The producer will have prepared answers and action plans for each of the comments. During the meeting the moderator leads the discussions, whilst the recorder enters the action and resolutions against each comment. Since the inspection has a time limit of only one hour, discussions on defect resolutions are taken as part of future actions to resolve a given comment. The objective is to identify as many defects as possible, not to solve them in the meeting.

• The recorder completes the inspection summary report using the tool and an official inspection exit form is signed by all attendees indicating pass, pass with rework, fail with another inspection required, fail with the inspection not complete.

• After the inspection the producer makes sure that all comments, meeting minutes, inspection exit form and follow-up actions are entered into the inspection tool. Other information is also entered such as who attended, status of inspection, duration etc.

• The hard copy signed versions of the inspection summary report and meeting minutes are then recorded in the QFSD repository inspection folder for that specific project.

• All open action items are then worked on until they are closed by the producer, this could involve a second inspection. When all actions are closed all reviewers are informed and can review the actions that have been carried out online. Once all are satisfied a new inspection summary report is generated and a hard copy signed by all reviewers and entered in to the QFSD repository inspection folder.

The inspection tool carries an electronic version of the full inspection procedure.
6.4.3 Technical design and deployment

The inspection utility is written using Microsoft Visual Basic™ and Microsoft SQLServer™ Database. The tool also communicates directly with Microsoft Exchange 2003™ (email), which enables inspection materials to be distributed and for inspection meeting dates to be automatically set-up in the Exchange Calendars of the invited participants.

The recommended use of the tool is to create a new inspection library per project and at the end of that project archive the whole inspection database in the project folder within the QFSD repository. In that way all project information is located in a single, version controlled repository, which makes future information access much easier for comparison of estimates or for audit purposes etc. The database is accessed via an API that can be run within a standard internet browser, which makes the tool accessible to all project members.

6.5 Clusterisation technique

Clusterisation is essentially a way of structuring the development and associated project management in order to decompose a software project into manageable sub-units, which improves quality and reduces overall risk.

Industry standard development methodologies such as the unified process can be used successfully with the QFSD framework. However, the Clusterisation methodology developed by the researcher as part of the QFSD framework has been used to deliver a significant number of successful projects at both Fisher-Rosemount and Celesio AG. The approach is incremental rather than purely iterative in nature. Clusterisation enables a software development project to decompose the development into a number of ‘vertical slices’ of functionality. Each vertical slice is assigned to a development sub-team and upon delivery of the clusters assigned functionality provides a working system that can be tested and released or form the input to the final release testing phase, which generally includes deliverables from all clusters in the project. In a typical project there are a number of clusters (or sub-teams), which include requirements, several for software production, installation &
user documentation, release testing and a general technology cluster that handles technical component upgrades or replacements.

One of the major differences in this technique is that a cluster is assigned a set of complete and operational user functionality to deliver, rather than implementing smaller pieces of the functionality. The idea is that when the cluster delivers the required functionality it is standalone and operational. Dividing the functionality between clusters enables a high level of parallel working to take place with all clusters delivering working functionality that can be tested and delivered to the customer. In order to achieve this high level of skill duplication a strong training programme is required, but the benefits in terms of more predictable deliveries and highly motivated developers is well worth the additional training effort.

The clusterisation approach is not simply a way of decomposing the development, the whole requirements identification, design, development and testing uses the clusterisation structure. This approach has proved to be very successful in both Fisher-Rosemount and Celesio AG, with development teams being able to deliver working functionality to the customer ahead of the final integrated product release.

Further benefits from this approach are that a working system is available earlier in the development process. In addition, project management and quality measurement and control is easier as the work in each cluster is more visible. For example all process performance quality measurements are based on measurements taken for each project cluster. Whilst the approach is not iterative, but rather incremental, it is a good first step for a new project group to attain control and improve the consistency of deliveries. A further benefit of the self-contained clusters is that they can be run in parallel or in series, plus not all clusters have to be completed in order to make a delivery of a working system to the customer.

The following is a very concise list of the benefits of adopting the clusterisation approach as taken from the relevant QFSD framework process description.

- Enables the development and all associated project management activities to be managed in smaller units. This makes the project less complex from both the development co-ordination and project management perspectives.
• As each cluster delivery provides an entire slice of working functionality, this can be tested and facilitate early defect fixing before going to final test. Provides a working reference system for other clusters to test their functionality against.

• Provides a working reference system for other clusters to test their functionality against.

• Each completed cluster could be taken as the final enhancement and allow the entire product to be moved to release testing. Note that in this technique all high priority customer enhancements are included in the initial clusters.

• Each cluster deliverable enables a working test system to be made available early for customer requirements validation.

• Project management is structured around each cluster and hence there is more transparency

• Performance and capability improvements can use the more granular information provided by planning and executing a project on a multiple cluster basis.

6.6 QFD software requirements capture and prioritisation technique

The capture and prioritisation of requirements is a major factor in ensuring that a software product meets the needs of stakeholders both in terms of functionality and availability for use. The researcher has found that using a modified version of the classical Quality Function Deployment approach (Day, 1993) to software requirements capture and producing formal software requirements specifications, complete with a unique requirements numbering scheme, to be the most efficient approach. This assertion is based on a considerable number of successful projects delivered by the researcher’s development teams in both Fisher-Rosemount and Celesio AG. The modified QFD approach is fundamental to the success of the QFSD framework as it not only provides a quality process for requirements gathering and prioritisation, but also feeds directly in to the detailed development and test planning processes.
Understanding the stakeholders prioritisation motivations also enables the development teams to deliver the higher priority requirements early in the project, which allows lower priority requirements to be moved to later in the project or to be removed from the current release if the project hits funding or staffing issues. Involvement of the stakeholders is ensured by their participation in the cluster correlation meetings, which require the stakeholders to actively participate in the requirements definition and prioritisation process.

Another major benefit of the QFD requirements capture and prioritisation process is that development teams and all other departments that support the software product are given an early opportunity to meet and discuss the requirements and what impact they will have on each department.

6.6.1 Overview of QFD requirements and prioritisation technique

The accepted best practice is that user requirements are crosschecked against the design (Wasserman, 1993), but the key is to have a method of holding each requirement against its equivalent release test case specification. This ensures that each requirement can be tested and verified as having been tested during the final release-testing phase. At first this would seem like an obvious approach, but keeping evolving requirements in-line with the test specification changes needs to be actively managed in order to avoid untested functions in the released software product.

A simple method to achieve this is to write a set of test case procedures (Test Procedure Specifications). It is important not to embed the requirements numbers against the test cases in these procedures, as this will prevent them from being used for regression testing in a later release. Instead, it is better to create a cross reference tracking spreadsheet or database, in which each test procedure, set of test cases, associated requirements and test results are recorded. Taking this approach ensures that all requirements are tested and it also enables test statistics to be generated and test cases to be reused as regression tests in later releases. Whilst this is really just good practice the question remains on how to determine which user and system requirements are really necessary to include in a software release and which are can be weeded out as wish list items.
The researcher has observed a number of companies who only gather requirements from its customers via warranty programs; customer complaints and sales staff feedback. Whilst this approach is a part of requirements gathering it tends to focus on what's wrong with the current products and not what will meet the customer needs.

The method used to capture customer needs within the QFSD framework is a substantially modified version of the quality function deployment methodology (QFD) (Anderson, 1996). The QFD methodology originated from the car industry in Japan and is now used in a number of manufacturing industries for the capture and prioritisation of new features within product lines. The method uses a hierarchy of matrices combined with weighting factors, which receive requirements input and associated criteria from all departments involved in the product manufacturing process.

In order to apply this to the capture and prioritisation of user requirements for software products a five level QFD matrix is defined. An example can be found in Appendix K. The five levels in hierarchical order are as follows (note each level has equal priority):

1. Company business objectives level
   - This ensures that the requirements contribute to the overall company direction and move it closer to its defined targets. This is key if the software product is to gain support from senior management and be seen as successful when it is delivered.

2. The product range development group's objectives level
   - Most software products are part of a portfolio of software products, either for market resale or within a companies overall IT landscape. Therefore all requirements must also contribute to the forward direction of the software product itself.

3. The configuration product group's goals level
   - This matrix is generally known as the software product enhancement matrix as it deals with improvements in the software products operation and addresses
such things as performance, usability, technical component replacements etc. However, in this example the researcher has assigned the name of a specific software product group in Fisher-Rosemount as the later explanation of the technique takes as its example a real project (Fisher-Rosemount Case Study Three)

4. Prioritised tasks matrix level
   - This matrix contains the major tasks and development milestones mapped against the prioritised delivery milestones

5. Actual tasks matrix level
   - This matrix forms the basis for the detailed development planning and includes all tasks, estimates, timescales, milestones etc mapped against the unique requirements identifiers

Each of the matrices above has a link to the matrix either above or below it in the hierarchy giving full requirements traceability through the development lifecycle.

The example in the following sections uses the Fisher-Rosemount configuration development product groups objectives matrix linked in to a single product range development objective 'develop a PC Windows based configuration product'. This objective is then linked in to a single company business objective 'sustain the current control system product line in the market place'.

6.6.2 QFD requirements and prioritisation technique structure

The following sections explain, using a real example, how the technique is applied. It should be noted that the tool used is a macro enhanced Excel template, which is located in the QFSD repository. When the template is fully populated and approved it is again located in the QFSD repository in the appropriate project specific folder.

The basic idea behind QFD is the translation of user requirements, expressed in their own, occasionally non-technical terms, into a set of measurable technical specifications. The translation is achieved using a number of matrices and tables, the
hierarchical nature of which enables a progressive refinement of the requirements; moving from general needs to concise technical definitions.

Once the requirements are understood the technique will yield a prioritised list in terms of user requirements and their correlation with the quality of the evolving product. In other words, it gives a chance to review all the proposed enhancements, apply a systematic approach to ranking them in which all stakeholders can participate.

For this example a QFD matrix (applies to level 3 in the previous section 'The configuration product development group's objectives') was created. The first step in creating a QFD matrix for a project is to define the component parts of a typical matrix (in this case a template from the QFSD repository was used), then tailor it to the particular needs / type of the specific project. Figure 6.3 shows an extract from the second Fisher-Rosemount Case Study QFD matrix.

![Example QFD matrix structure](image)

Figure 6.3: Example QFD matrix structure

Once the QFD matrix is completed, it will provide a ranking for each of the configuration product development groups goals, based on the correlation assigned between each user requirement and the project goals. This is balanced by the weighting factor generated in the customer demand column i.e. weighed input of the
customer demand, which is provided by other departments such as marketing, sales, support etc.

When all requirement requests have been correlated and the goals have been ranked, a cross departmental review group then prioritises the requests initially as being in the release or not. Using the requests that are deemed to be in the release, priorities are assigned for their implementation order (1, 2 and 3). The requests are then assigned to project milestones and cluster groups (see Section 6.5). In addition, each request is assigned to a Software Requirements Specification and is given a unique requirements identifier. This enables traceability between the request and its formal requirements definition and identification.

6.6.3 QFD matrix components

The following provides more detail on each component within the QFD matrix analysis spreadsheet, which is based on the Fisher-Rosemount Case Study Three.

6.6.3.1 Project Beneficiaries

The first step in the QFD process is to identify the beneficiaries of the project. The beneficiaries are typically identified by the project and marketing product managers and include:

- Installed Base customers who use the existing software product
- New customers especially if this is a new software product
- Value add reseller offices & internal users
- Product marketing & sales
- The product development team
- Customer support teams
- Computer operations teams

Once the project beneficiaries have been identified, the initial requirements for the product are collected from them using the following technique:
A detailed and extensive questionnaire, which can be found in Appendix L, is distributed to each of the beneficiaries in the form of a Web-based questionnaire. The questionnaire provides details from the previous two QFD hierarchies together with a concise statement of each of the proposed requirements. The results from the questionnaire are gathered and analysed against the set of key design goals. The design goals are established by the software project and product managers (or operational business stakeholders) based on the previous two QFD hierarchies: company business objectives matrix and the core product development group’s objectives matrix.

The results from this analysis form the raw materials for the cluster correlation meetings.

6.6.3.2 ‘What’s (Requirements):
These are initially provided in the ‘beneficiaries’ own words, but are then refined to be suitable for the requirements component parts of the matrix during the cluster correlation meetings i.e. translated into precise brief requirements statements.

Other sources of requirements that are typically factored in to the requirements capture are technical support product tracking database and feedback from the previous releases customer field trial phases.

6.6.3.3 ‘How’s (Goals):
The ‘How’s are, in this case, interpreted as the goals of the project.

Table 6.1 gives an example list grouped by development cluster, which takes in to account the next process ‘correlation’. The headings relate to cluster names from an example project.
Table 6.1: Cluster Goals

<table>
<thead>
<tr>
<th>Server Engineering</th>
<th>User Interface</th>
<th>CDOS</th>
<th>Snap-On</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daemon Deadlock Issue</td>
<td>Tag Containers</td>
<td>CD P1.0 to CD P2.0 Upgrade</td>
<td>Display Designer</td>
</tr>
<tr>
<td>Add User Defined Bytes</td>
<td>Interactive Validate</td>
<td>ENVOX P3.4 to ENVOX P4.0 Upgrade</td>
<td>User Defined Functions &amp; Limited Legacy</td>
</tr>
<tr>
<td>PV Failure Attribute</td>
<td>IDI Card Backup</td>
<td>CDOS / ENVOX P4.0 installation</td>
<td>FST</td>
</tr>
<tr>
<td>Set point Velocity Limit</td>
<td>Multi Level Tune</td>
<td>ENVOX P4.0 VAX and AXP installation</td>
<td>Simple Text Box</td>
</tr>
<tr>
<td>Over / Under Range</td>
<td>Audit Trail Utility</td>
<td>Control Desktop P1.0 Compatibility</td>
<td>Full Multiple Resource Support</td>
</tr>
<tr>
<td>AVP &amp; Alerts</td>
<td>Matrix View for Loops</td>
<td>CDOS...ODBC Compliant</td>
<td>ASCII Import</td>
</tr>
<tr>
<td>IVP Hold Attribute</td>
<td>Upgrade to PB 5.0 or 6.0</td>
<td>Increased TDS Packet Size</td>
<td>Support for Comm / space</td>
</tr>
<tr>
<td>Multi Levels of Tune</td>
<td>Navigators - replacement of</td>
<td>CDOS Performance 1</td>
<td>Right Mouse</td>
</tr>
<tr>
<td>HP Spin=2</td>
<td>Console Relationships View</td>
<td>Scope by Data Window</td>
<td>Menus in FBE</td>
</tr>
<tr>
<td>ENVOX Issues</td>
<td>Create From Data</td>
<td>CDOS Replay/online tables</td>
<td>Diagnostics</td>
</tr>
<tr>
<td>ENVOX Diagnostics</td>
<td>Points not done in P1.0</td>
<td>DB Lib connect limit</td>
<td>Indirect Access to (IO, Point Atts, Mem Var)</td>
</tr>
<tr>
<td>HDL Rev Data</td>
<td>Matrix Config F.C.C</td>
<td>List handling (group templates)</td>
<td>ODB on NT</td>
</tr>
</tbody>
</table>

6.6.3.4 Correlation Process:

The next stage is to establish the correlation between the How's & What's i.e. the beneficiaries requests and the goals of the software product as a whole. For example, a user requirement may provide a high productivity gain, but could impact the stability of the product if implemented with current technology. Correlation will also contain some 'soft' information i.e. how the change would be accepted by the organization as a whole. The correlation values used are: 9,3,1,0 (Strong, Medium, Weak, No Correlation). These correlation factors are taken directly from the classic QFD method (Day, 1993).
The correlation process is carried out by first translating the beneficiaries' needs statements into concise 'what' statements.

The next step in the process is the mapping of each of the 'what's' to a specific development cluster. A series of correlation meetings then take place where each cluster group meets with the project and product managers (or operational business stakeholder), to review each 'what' and assign correlation factors. It needs to be recognised that the completion of the matrix is not the main benefit of the QFD process, but rather the process whereby the team and management discuss each of the beneficiaries' requests. This process enables the developers to be involved in the early stages of customer requirements capture, which promotes understanding and buy in. It also enables the development team to understand the development tradeoffs that could be made during the later project execution-planning phase. In addition, notes are held against each row in the QFD spreadsheet, which record the major agreements and discussion points that were raised at the correlation meetings. The results from the correlation meetings are used to generate the QFD matrix, containing the correlations between the 'what's' and the 'How's.'

6.6.3.5 Importance to the user (I)

The next step in the process is to provide product marketing and the customers with the opportunity to weight the 'What's. Each 'What' is examined with respect to its importance to the user and an assessment of its relative weight applied. This process is carried out at a number of customer technical steering committee meetings, where customers were asked to rank each 'what' using the values 1 to 5 (1 being low, 5 being high in terms of customer importance).

6.6.3.6 Sales Points (S):
The sales group are given the opportunity to rank the 'what's by their sales impact. The sales points values used were: 1.5, 1.2, 1 (1.5 Important, 1.2 Less important, assign (1) to all the rest). The sales points are defined as the value a feature would have in promoting sales of the product in the market.
6.6.3.7 Performance Ratio (R):
Each 'What' is compared by the project and marketing product managers, against the current performance provided by the existing product and performance improvement provided by the proposed enhancement. This is expressed as a decimal fraction i.e. 50% improvement = 0.5.

6.6.3.8 Ranking the 'What's:
Once the basic absolute weights are assigned, the normalized weights are then calculated. This results in a set of prioritised enhancements (what's), together with their correlation to the main goals of the project, on a per cluster basis. The next step is to work out the weighting factors for each of the 'What's, from a customer perspective. The following gives details of the calculations used:

- The absolute weight of each 'What' \( W = I \) (importance to the user) \( \times S \) (Sales Point) \( \times R \) (Performance Ratio). This gave a ranking in terms of the customer, marketing and sales.

- The absolute weight column \( W \) is then normalized to create the user demand ranking column, which is \( D \).

6.6.3.9 Ranking the How's
The correlation values in each 'What' row are multiplied by the corresponding value in the user demand column \( D \). The resulting products in each 'How' column, are summed to obtain the absolute weights. The absolute weight column is then normalised to rank the project 'How's.

6.6.3.10 Final enhancement ranking, prioritisation and assignment:
At this stage the QFD matrix spreadsheet (with the calculations already embedded) contains the 'what's listed in customer priority order and the 'How's listed in customer, development team, marketing and sales priority order. The QFD matrix spreadsheet highlights how each requirement in a specific project area, is mapped against user (used in its broadest sense) demand, project and company goals.
The next stage is to generate the overall prioritisation of the ‘What’s for assignment to implementation development clusters.

The process used is as follows:

1. The user demand column (D), in the matrix, gives the basic user demand for a particular requirement. The N.weight row put the ‘How’s, or project goals, into a priority order. From this information the project team, project & marketing product manager prioritised the ‘what’s list (using priorities 1, 2, or 3 ) by looking at the user demand level and how it fits with the project goals. For example, a requirement may have a high customer demand, but does not fit in with the projects goals or fits in with a goal, which has been ranked as a low priority goal. The ranking of the project goals provides feedback on the thought processes (or prejudices) used when assigning the correlation values.

2. Each requirement is assigned a priority 1, 2 or 3.

3. This QFD spreadsheet is then represented to the stakeholders for final approval.

4. Once customer approval is obtained, the requirements are assigned to implementation clusters and given a delivery milestone. Whilst the dates for the milestones have not yet been set, this assignment is the first step in the project execution planning process.

5. The requirements are assigned to software requirements specifications. Each development cluster produces its own requirements specification.

6. Following the initial project planning selected requirements are transferred to the task monitoring spreadsheets for each cluster. These are used to plan the activities required in order to delivery the requirement.
6.7 Defect rate quality and release indication technique

6.7.1 Overview of defect rate quality and release technique

One of the major challenges associated with release testing is to determine when the product is sufficiently stable for release. The method developed for inclusion in the QFSD framework is based on a defect rate prediction technique.

The decision to develop this approach further in the research came after reading several publications on the typical defect discovery rates of software products deployed in the market (Wood, A, 1996). As might be expected, software products such as stable versions of the UNIX operating system had infrequent incidents of high priority defects. However, products such as Microsoft Word™ will have a significantly higher defect discovery rate. However, both products meet the needs of their customers.

This shows that while every software product should be tested across its whole functional range, a defect discovery rate can be set in order to determine its stability for release to any given market. For example, if the product is a desktop publishing application, then carrying out a set of full regression tests over a five day period, with no critical defects being discovered is a reasonable release criteria. This, of course, makes the assumption that the desktop is running on Microsoft Windows™ and that the PC will be rebooted at least once in any five day period and hence will release memory etc.

Basically, the philosophy of testing in the QFSD framework is not to test until zero defects are achieved, which is not really possible, but one of a strategy aimed at measuring an acceptable error rate level for each of the software components. This statistical method based on measuring a predefined defect rate below which a product is acceptable, is both practical, and very cost efficient.

6.7.2 Application of defect rate quality and release technique

The first step in the technique is to establish what the exit criteria are for releasing the software product from the testing phase. This will depend on the particular software
product. I.e. is this commercial software or safety critical software? Obviously there are many levels between these two extremes that can be established. As an example the following exit criteria were used in a control system engineering workstation software product, the application of which was safety critical, but not to the extent of say aircraft or nuclear software based control systems.

- **First criterion:** Zero 'critical' defects (loss of functionality or extreme usability).
- **Second criterion:** Less than 5 'serious' defects (no loss of functionality, minor usability problem, workaround available) and less than 20 'cosmetic' defects (minor visual or operational variance). The defect rates used in this specific project had been systematically reduced over a number of releases as the product became more mature and testing techniques were improved.
- **Third criterion:** Satisfactory results for the products Beta Test and Manufacturing Pilot run.
- **Fourth criterion:** A defect rate of 0.01 serious defects per hour during full testing for a period of seven days.
- **Fifth criterion:** A defect rate of 0.1 defects per hour for all defect levels during full testing for a period of seven days.

The above criteria assume that all major requirements have been implemented and that the entire product has been tested, both regressively and in terms of new functionality. The information on defect rates captured during the testing phase is presented graphically for each major software component. The publication of this information on a project web site can be used to encourage developers to drive down defect rates. This approach makes testing part of the overall quality process rather than an end of project error trapping exercise.

The following sections show the major defect rate graphs used and a sample of the individual software components rate graphs taken from the Fisher- Rosemount Case Study Three project.

### 6.7.2.1 Overall testing hours graph

The graph shown in Figure 6.4 is required to enable the defect rates to be interpreted correctly. It shows that the testing hours are not dropping towards the end of the
project, in which case the target defect rate would be achieved in a meaningless way as the number of defects discovered would reduce naturally as testing hours reduced.

![Total CDP 2.0 Test Hours](image)

**Figure 6.4:** Overall testing hours' profile

Figure 6.5 shows the total numbers and classifications of defects discovered during the whole testing process. The dip in the defect discovery rate during week beginning 24\textsuperscript{th} April is interesting. Remember this graphic is a summary graphic of all individual clusters testing activities and hence represents a general characteristic. The researcher has observed this characteristic in previous and subsequent release testing phases for this software product. However, this characteristic can be explained easily as the testing approach was split into two parts.

The first part was the execution of regression testing, which includes automated testing scripts designed to flush out defects that have been introduced in existing code that has been modified during the specific project. Once the regression tests have been executed, defects fixed and regression tests re-executed the defect rate discovery shows a drop to levels set by the exit criteria. During the next phase of testing the regression testing suit is run automatically overnight just in case any of the further defect fixes in the new code for the release have a knock-on effect on existing code.

The second part was the execution of new test plans designed to test the new functionality added to the product for this release, plus a programme of ad-hoc user
and overnight performance stress testing. As can be observed this second wave of testing started to raise new defects as expected.

Figure 6.5: Total numbers and classifications of defects discovered

6.7.2.2 Predicted software component defect rate

The researcher has observed that over many individual projects and within each project cluster, the defect discovery rate follows a 3rd order polynomial as can be seen in Figures 6.6 and 6.7. This trend can be used to predict the likely duration and effort required to meet the testing exit criteria. Obviously this is easier if the product under test is mature and hence the overall testing effort and durations are available from past projects. However, for a new product the approach is to try and make the testing hours executed each week constant and monitor the defect rates until they reach the top of the initial curve, usually around four weeks for a large software change. Based on this, the polynomial curve can be used to predict when the defect rate is likely to reach acceptable levels for release.

Figures 6.6 and 6.7 show the defect rate information based on the Fisher-Rosemount Case Study Three. Figure 6.6 shows cumulative defect rates for all clusters within the project, whilst Figure 6.7 only shows the critical defect rates. Both
diagrams have superimposed the actual calculated polynomial curve calculated using the defect rate results.

The polynomial predicted that the project would reach the release rate of 0.01 defects per hour for critical defects in test week 11. This was the case with the diagram showing the rate being driven to zero and held there for 7 days.

![Faults Per Test Hour Overall](image)

Figure 6.6: Total defects discovered rate graph
6.8 Conclusions

6.8.1 The online QFSD electronic management repository

This chapter has shown that the QFSD framework is supported by a modern, flexible web-based repository tool. The tool provides the structure, materials, processes and controls that are necessary in order to establish a high quality software delivery environment. The framework also provides techniques for improving development, requirements capture, inspections, testing together with a built in quality performance monitoring tool and capability analysis.

In addition, as an IT development organisation becomes more experienced then industry standard techniques and tools can be integrated with the central QFSD repository. For example the Rational Unified Process (RUP) and its associated tooling, and the Prince2 project management templates can all be integrated. However, whilst other industry standard tools and techniques can be integrated the QFSD framework remains the master document and procedure source repository.
6.8.2 Process performance software package and capability database

Bach (1994) makes the observation that both the CMM maturity and the SEI (Software Engineering Institute) models are themselves immature and appropriate only for large projects in the defence industry. The main reason given is that these models are created and supported by people in those defence industries. On the whole, the view stated is that these models provide a set of generalised software processes that do not match the way software professionals actually work.

One of the key strengths of the QFSD framework can be seen by considering the process performance and capability features. In the framework it is recognised that software development processes must be monitored and adapted in order to deliver high quality software, but also to deliver that software to a given market, with a given quality level, in an acceptable timeframe and at an acceptable cost. Therefore the framework both promotes and fully supports a continuous monitoring and review mechanism to ensure that the processes defined in the framework can be quickly modified in order to optimise the overall software delivery process.

6.8.3 Inspection utility

Another key process in the QFSD framework is the inspection process, which is based on the method developed by Michael Fagan. Using formal inspections together with the custom tool developed as part of the framework is one of the key element in promoting and monitoring the adoption of a quality culture. With out the application of inspections at every phase of the development lifecycle there is no foundation upon which a quality software development environment can be built.

Doolen (1992) states that since the introduction of formal inspections at Shell Oil, which have trapped one third of defects at the requirements stage, there has been a significant saving in development effort by not having to find and fix the defects at later phases in the development.
6.8.4 ‘Clusterisation’ technique

One area that is missing from quality frameworks in general is an overall development methodology. Whilst there are many development methodologies available from which to chose, the link between the quality framework and the particular framework is left up to the user to figure out. Whilst the QFSD framework can be used with other development methodologies, it is delivered with its own development method known as ‘Clusterisation’. Whilst Clusterisation has been described earlier it is worth highlighting the fact that the methodology is designed with three overriding objectives.

1. To provide a methodology that minimises the risk of non delivery by using pragmatic problem decomposition approach.
2. To ensure that the methodology is integrated closely with the quality processes within the framework enabling easy monitoring and improvement.
3. To make sure that the methodology is simple to learn and apply.

6.8.5 QFD software requirements capture and prioritisation technique

It is generally accepted that the capture and definition of user requirements is an area of potential risk in the success of any software project. The researcher has observed this failure in a recent large project within Celesio AG, where in order to show progress early, the user requirements were not fully reconciled with the end users. The impact has been a twelve months slip in the project in order to address a significant number of change requests.

The QFSD framework places major emphasis on the capture, definition and prioritisation requirements with the direct involvement of all stakeholders through the use of its modified QFD process. This process also ensures that the project is fully aligned with the overall objectives of the business in order that maximum ROI can be achieved.
6.8.6 Defect rate quality and release prioritisation technique

Jones (1995) makes the point that by not having formal inspections and relying on finding defects during the testing phase, the duration and effort required to execute the tests and fix and retest the defects makes the delivery schedule unpredictable.

The QFSD framework addresses the first part of Jones' statement by adopting a formal inspection technique and applying it to all stages in the development lifecycle. However, this does not in itself give predictability in the test phase duration and effort. For that we need to consider what the exit criteria are from the test phase and for this an acceptable defect level (or rate) needs to be established. It should be understood that whilst too little testing results in additional effort and cost, the same is true, but to a lesser extent, for over testing as it is creating a level of quality that is over that required for the specific target market.

To address the above the QFSD framework provides a defect rate target and monitoring technique, which has proved very effective in both Fisher-Rosemount and Celesio AG.

6.8.7 Utilisation

This chapter has described the major tools and techniques that have been developed in order to support the application of the QFSD framework. It should be understood that these tools and techniques, as described, underpin the case studies and further framework process descriptions that follow in subsequent chapters.

It should also be realised that the platforms and technologies on which the tools are built have been updated in order to take advantage of new technologies. For example the repository has been deployed using Microsoft SharePoint™ and plans are in place for a future version using SAP Portal™.
Chapter 7 - Case Study Three: Validation of the core QFSD model at Fisher-Rosemount

This chapter describes the application of the core QFSD framework as described in Chapter 5, to a new software product development at Fisher-Rosemount. The Chapter also provides a case study (Fisher-Rosemount Case Study Three) summary explaining how successful the application of the new framework was to the project considered within the case study.

The detailed application of each key framework process used in the case study can be further referenced in Appendix C. This provides a complete reference to the frameworks processes. Prior to the case study, the core framework had been refined and tested over a number of previous projects within Fisher-Rosemount. However, the decision to use this project as the formal validation was taken to ensure a like-for-like comparison, due to it having a similar nature to the first Fisher-Rosemount Case Study One described in Chapter 4.

The first objective of this chapter is to show that the basic core QFSD framework can be applied to a software project and result in improved quality and predictable delivery timescales. However, this is focussed more at the software practitioner and therefore the detailed application of each process during the case study and its direct project benefits can be found in Appendix M. The second objective is to introduce the reader to the principles and rationale for the application of the processes within the framework. An understanding of these principles will help in subsequent chapters which build upon the core framework. Figure 7.1 shows the elements of the QFSD framework that are validated in this chapter.
7.1 Introduction

Figure 7.2 shows the overall structure of the QFSD framework as applied to this case study.
For this case study, the prototype QFSD online repository was used and made available via a web browser to the project team involved and to stakeholders in the business. A full description of the repository design is given in Chapter 6. The repository has two separate areas. The first contains the core QFSD processes descriptions, plus a set of document templates for the project artefacts. The second allows a standard navigation structure to be created that is specific to the project and in which the project artefacts are stored and published. The repository had, at the time of the case study only basic capability such as version history, change history and check-in and check-out. The navigation structure could be viewed graphically and the documents viewed or checked-out for modification.

The overall project management plan template document was created and populated by the project manager. The detailed structure of the project management plan template is shown in Appendix E. The core QFSD processes, which are a refined version of the ones used in the first Fisher-Rosemount case study, can again be referenced in Appendix C.
Fisher-Rosemount Case Study One concerned the development of a new control system configuration product. In order to get an improved validity level between the case study results, Fisher-Rosemount Case Study Two was based on a new development project to replace a substantial part of the product developed during the first case study.

To provide a practical example of applying the QFSD framework, the processes contained within the organisational and management areas of the model will be explained in terms of how they were applied during the case study. The supporting and implementation processes have been omitted as the information required to describe how these were applied would be extensive and can be referenced for the case study in Appendix C, which contains the case studies project management plan. In addition, the majority of the implementation processes have subsequently been replaced by the adoption of the universal process (UP) into the framework as an optional extension.

The project's main objective was to develop a PC Windows client and POSIX compliant server (e.g. VAX VMS, Open VMS, IBM AIX & HP-UX server) configuration product to supplement the current VAX VMS workstation based application. The configuration application is used by either the company's own engineering departments, VARs (value added resellers) or customer's engineers, to describe how a process (such as drug manufacturing recipe) will be configured, controlled and monitored by the target process control equipment.

The development approach used in the case study is not based on the unified process (UP), as at the time UP was very new and by no means a complete or proven methodology. Instead, one of the core QFSD methods developed by the researcher was used called 'clusterisation' (refer to Chapter 6 for a description of this technique).

As a first step, and in order to reduce development risk, the new product development was divided into three separate product releases, as follows:

1. Replace the client functionality of the existing product by replicating its features on a PC Windows client. These new replacement clients were not exact replicas of the
old workstation clients. The development took advantage of the user interface power provided by the Windows environment. The new clients worked seamlessly with the existing customer databases, which ran on DEC VAX, DEC Alpha and HP-UX, P-series IBM and IBM Power PC workstation platforms.

2. Use feedback from customers on the previous released configuration product and improve the client functionality providing best-in-class productivity improvements.

3. Port the database and associated processing to a multi-platform 'POSIX' environment. Transfer of customer's databases from the DEC workstation platforms to the new open platforms would be carried out by a simple transfer and upgrade utility.

The case study, refer to Appendix M, shows how the first release above was planned, executed and successfully delivered using the new quality framework. It should be remembered that all QFSD processes and artefacts are handled online and under configuration control for this case study. The remaining two releases were subsequently successfully delivered using the QFSD framework.

7.2 Conclusion

This Chapter and Appendix M has shown how the core QFSD framework methodology can be applied to a major new software product development with a high degree of success.

In order that the key planning, monitoring and control processes are conveyed to the reader, the case study concentrated on presenting the organisational and management processes primarily. In addition, key techniques for requirements validation, prioritisation and planning were also introduced.

There are strong indicators, following the results from this case study that the framework processes can be successfully applied to a complex new product development. The new software product was delivered on time and within budget and was generally well received by the customers. A revealing illustration of this is shown in Figure M2, the staffing profile for the project. Unlike the equivalent in the first case study there was no panic drafting of large numbers of extra personnel to try to keep
the project on track. The profile shows that software projects can be properly planned, managed and resourced if the business objectives and critical requirements are kept as the main focus.

This chapter has given details of how one new software development project was successfully executed. The principles, tools and techniques and prescriptive nature used in the framework are applicable to a wide range of software projects. This assertion is supported by the framework's successful adoption at both Fisher-Rosemont and Celesio AG who have deployed the framework on a significant number of projects. This framework, therefore, should be a basis upon which any organisation can develop successful quality software. In order to support this statement a further case study (Case Study Four) is used in Chapter 8 to show the actual level of quality that is achieved by using the QFSD framework.
Chapter 8 - Process performance and capability technique

This chapter addresses the QFSD process performance and capability additions to the framework. Whilst having a framework covering an extensive number of development processes, process performance must be measurable in order to determine if the processes being applied in the framework are resulting in improvements. Process performance is a measure of improvement in the underlying result of applying the framework process. If not, they must be modified. In the same way, the overall competence of the actual project team's application of the framework processes must be measured in order to measure the project team's effectiveness. This measurement is defined as process capability. A combination of both process performance and capability is needed in order to evolve both the processes themselves and the efficient application of the processes by the project teams. Figure 8.1 shows the components of the QFSD framework that are validated.

Figure 8.1: Framework validation and progress chart - process performance and capability
8.1 Introduction

Whilst there are a number of candidate methodologies that could be considered for integration into the QFSD framework in order to provide a measure of both process performance and capability, the classic being the Capability Maturity Model (refer to section 2.4). The methodologies did not fit the objectives profile for inclusion in the framework.

Experience at Fisher-Rosemount and Celesio AG has shown that the following are key attributes for the successful adoption of a process performance and capability methodology:

- Must be simple to adopt by both project management and development teams
- Must become an integral part of the project methodology using measurements that are readily available requiring no additional effort for their generation
- Must provide visible process performance and capability monitoring both within projects and across projects

A review of existing methodologies was carried out in Chapter 2 and highlighted weaknesses in terms of complexity of adoption, requirement of additional metrics outside of the normal project management and development data, plus a weakness in terms of lack of performance measurement. Therefore, a statistically based process performance approach together with a simple process capability measurement and improvement scheme was designed for QFSD.

Given that the underlying principles of the process performance and capability methodology have been described in the Tools and Techniques Chapter, this chapter will describe the practical application of the methodology using a real case study (Case Study Four). The case study describes the application of a statistical process performance monitoring and capability assessment on a large process control software project. The process performance and capability monitoring used in the case study is a fundamental part of the QFSD framework. Both the process performance and capability statistics are tracked in a database, which is accessed via the QFSD online repository. The process performance measurements were
based on commonly available metrics that could be obtained with the minimum disruption to the processes being examined. The application of statistical methods was used to establish problem areas at the earliest opportunity allowing process adjustments to be made to improve the process performance.

The processes used were defined in terms of work products, which were categorized and evaluated for the level of completion. This allowed a process capability to be calculated. Those processes, which were found to be at a lower capability level, became the focus for the process improvement for the next project. In particular, the capability measurement identified processes where questions needed to be raised about the relevance of the process, whether the techniques used were effective and whether the tools were adequate.

It was found that achieving high performance and capability is a learning process with the development team improving with each new project. The benefits have been better managed, more cost and time effective projects producing higher quality software. Management and the development teams obtain a better understanding of the software development process and this continuing learning process leads to a continuous improvement in both the development methodology and the resulting software. This chapter presents examples of the real benefits that can be obtained by tracking process performance and assessing process capability at all stages, and shows that this can be achieved without the need to resort to complex procedures for process measurement, as most of the empirical data came from commonly available process data.

8.2 Process performance measurement methodology

An analogy can be drawn between continuous process improvement and the tuning of a control loop. When a development attains ISO9000 accreditation, the loop control can be thought of as ‘open loop control’. Open loop control is where the action of the system is independent of the control system output i.e. a software development is applying procedures but not using any feedback mechanism to ‘tune’ the processes in order to continuously improve them. By the application of suitable metrics, a development can turn the loop control in to ‘closed loop control’. Closed
loop control is where the control action is modified based on the control system output. This control loop analogy is presented in Figure 8.2.

![Diagram of Loop Control](image)

Figure 8.2: Open verses Closed Loop Process Control

The aim of the QFSD performance methodology is to monitor, on a sample basis, project performance and hence improve project management control and the development processes. However, this must be achieved with the minimum of disruption to the project processes being monitored.

The first step is to identify the points in the development when metrics should be gathered. These points have to be meaningful and be unobtrusive in terms of the actual development activities. A distinction must be made between basic project monitoring, problem resolution and the collection of metrics for subsequent analysis. Burr (1996) suggests that metrics need to be taken at least once a month. Whilst this is not disputed, it is necessary to ensure that the frequency of taking metrics does not impact the ongoing project work. Development teams will resent having to stop direct development tasks to provide metric information. A balance needs to be struck between the project manager's need to collect metrics and the development team's ability to provide them. The approach proposed suggests that the gathering of metrics should be carried out continuously, but the analysis should be carried out at the various strategic milestones in the project.
The metric information needs to be available as part of the development teams normal planning, execution and estimating activities. The presentation of the gathered metric information needs to be graphical and easily accessible by the project manager and the development team. The approach taken is to present the metrics analysis results as part of the projects QFSD intranet web site. The increased availability of the metrics analysis results increases the acceptability of the metrics gathering process by revealing the benefits and allowing the development team to feel they “own” the results.

In this case study project, the metrics were collected at each of the following project stages:

1. At the end of each project milestone. This enables the planning information for the milestone to be compared with actual results and therefore provide a basis from which to adjust subsequent development milestones.

2. During release testing. Testing metrics need to be continuously monitored to determine the stability of the product and to focus areas and types of testing.

3. Prior to the decision to release the product. This will enable the release criteria to be judged more accurately.

4. Following the product release during the project post-mortem. This will enable the quality process applied to the development to be reviewed with respect to process capability, and improvements to be scheduled in to subsequent projects.

From the above it is clear that the metrics that are gathered are different for each project phase. For the project in the case study the following sources of metric information were used:

1. The project management plan, which specifies the QFSD processes selected for application to this case study project.

2. The Microsoft Project™ schedule providing both estimated and actual time and resource loading data.
3. Task breakdown, estimate and progress monitoring spreadsheets giving a more detailed breakdown of time loading.

4. Estimated and actual resource loading on a per milestone basis.

5. The master project requirements list generated as a result of the quality function deployment (QFD) requirements validation process.

6. Defect rate monitoring information.

The above provide a good source of metrics and are all easily available for a typical project.

More refined metrics which pertain to the code development can be applied. However, metrics need to be applied progressively, therefore the above set was chosen as they are not obtrusive and the majority of the metrics were already being recorded in some fashion by the development group.

8.3 Process performance analysis methods

The QFSD performance methodology requires metrics to be recorded and analysed at the project stages indicated in the previous section. The approach taken is to record project estimate and target information at the start of each milestone for the current and all subsequent milestones.

At the end of each milestone, the actual performance data is recorded and compared with the milestones estimates and target information. Both the estimated and actual information is then recorded in a Control Diagram (Burr, 1996; Shewhart, 1931) database, which enables the performance of the processes to be monitored at each milestone graphically, and for process inefficiencies to be identified early and corrected.

Control Diagrams enable significant variances outside the norm to be highlighted and action taken before the next milestone. The action will involve analysing the reasons for the unusual result to either correct an under performing process or to learn from a process improvement to see if the lessons can be applied elsewhere.
The analysis may find either that the significant data may result from the process implementation or from the process estimation but in either case action can be taken based on the findings. An example Control Diagram is shown in Figure 8.3.

The end of milestone analysis of the Control Diagrams is the key to the identification of processes that required attention. Following the analysis, the knowledge gained is applied to refine all subsequent milestone estimates and processes. In this way, the processes and performance estimates are refined continuously throughout the project.

![Control Diagram Example](image.png)

**Figure 8.3: Control Diagram Example**

In the case study, a Control Diagram was created for each project cluster. The project clusters, against which baseline information and actual performance results are recorded, were as follows: server engineering, user interface, CDOS, SNAP-ON, requirements capture, test design, test execution, inspection and defect repair, where SNAP-ON and CDOS are the names of two major project components. The control diagrams produced for this project monitored the performance of each project cluster for each milestone. The control diagrams highlighted areas, which were performing outside control limits and required action. The initial control limits are calculated from historic data in order to determine what the natural variations in the specific item being measured are generally. As improvements in accuracy are made the control limits can be artificially restricted in order to drive better performance. The
measurement outside the control limit was then used to trace the source of the problem, starting at the cluster level and drilling down through the project task tracking, schedule information, software requirements, design and even down to the code line.

The control diagram used to determine if a process was predictable or to see how changes affected the process is known as a variables control diagram. Variables data was analysed in pairs of charts, which present data in terms of location or central tendency and spread. Location shows data in relation to the process average. Spread looks at piece-by-piece variation. The type of control diagram chosen for the data analysis was the X-Bar and S chart (Burr, 1996). This type of chart creates a picture of a process over time.

For each process, the estimated and actual times or other resource loadings were recorded for each sub task of the process. The X-Bar chart shows the average or mean of each subgroup of data. The S chart shows the standard deviation of each subgroup. The standard deviation is used in preference to a simple range plot, as the subgroups can contain 10 or more data points.

The X-Bar Diagram calculations were based on the following, where \( \bar{X} \) is the average of the estimated and actual time for each sub task and \( N \) is the number of sub tasks.

The central line average (CL) = \( \bar{X} = \frac{\Sigma \bar{X}}{N} \)

The standard deviation (\( \delta \)) = \( \frac{\Sigma \delta}{N} \) where \( \delta = \sqrt{\frac{\Sigma (X-\bar{X})^2}{N-1}} \)

The upper and lower control limits are as follows:

\[
\begin{align*}
\text{UCL}_{\bar{X}} &= \bar{X} + A3\bar{S} \\
\text{LCL}_{\bar{X}} &= \bar{X} - A3\bar{S} \\
\text{UCL}_S &= B4\bar{S} \\
\text{LCL}_S &= B4\bar{S}
\end{align*}
\]

These formulas usually use tabular constants based on sample size to calculate A3 and B4. A & B are indices based on sample size. Note that A3\(S\) is 3 standard deviations which equates to a normal distribution. The actual software used to
generate the Control Diagrams for the case study uses estimated sigma (not look up tables).

To generate the Control Diagrams a database was created using the statistical analysis software ‘SQCpack™’. This software package enables the cluster data to be added to the database and graphical X-bar and sigma control charts to be generated. The database is divided in to a number of data groups.

Each data group represents a project cluster. All data groups have a Control Diagram which is updated at the end of each project milestone. The Control Diagram data points were built up from the individual averages for each cluster and milestone data group. A simple percentage technique was used to generate individual values, which represent accuracy or conformance, depending on the information being gathered.

If concerned with estimating, the accuracy for an overestimated performance became:

\[
\%\text{ accuracy} = \left(\frac{\text{Actual Performance}}{\text{Original Estimate}}\right) \times 100
\]

For an underestimated performance the accuracy was:

\[
\%\text{ accuracy} = 100 - \left(\frac{\text{Actual Performance} - \text{Original Estimate}}{\text{Original Estimate}}\right) \times 100
\]

For information not directly concerned with progress (e.g. When testing, the number of tests passed first time), the definition of % conformance was:

\[
\%\text{ conformance} = \left(\frac{\text{number of conforming items}}{\text{number of possible items}}\right) \times 100
\]

As each value in the database was, itself, an indication of the percentage accuracy of an individual estimate, the values taken for a complete milestone displayed on a Control Diagram the natural cycle of the estimates as they pivoted about the mean. Any values outside of the UCL and LCL limits were then candidates for further investigation and improvement. However, this implies that the mean value was itself acceptable. If the mean value and its associated UCL and LCL limits were all outside the accepted range, then the root cause needed to be found as it indicated a major problem with the process or misapplication of the process.
There were four steps applied when carrying out the Control Diagram analysis.

1. Check to see if the central line, CL, the upper control limit, UCL, and the lower control limit, LCL, were within the expected and accepted range. If not, then the whole process or its application needed to be reviewed.

2. Identify all points that were outside the upper or lower control limits.

3. Identify any unusual pattern or trend. There can be number of unusual trends e.g. a sinusoidal pattern can indicate a cyclic or reoccurring problem.

4. Identify points in the middle third zone of the distance between control limits. 2/3rds of all points should occur in this middle 1/3rd zone. The more points that are in the middle 1/3rd zone the better the process. Whatever is causing this improvement needs to be maintained. This is based on the properties of the normal curve, which shows that the area under the curve between plus and minus one standard deviation is 68%, i.e. about 2/3rds.

A second type of diagram was used to examine the normal distribution of actual values. This was not the S chart discussed earlier, but a histogram presentation, which made the information easier to interpret. The number of values (e.g. % accuracy values) in intervals at different standard deviations round the mean was plotted. This then highlighted the individual estimates and measures that needed to be investigated as a result of step 4. An example of this type of diagram is given in Figure 8.5.

8.4 Actual Milestone Control Diagram Measurement Results

The case study project was executed using the incremental development process option. The project was based on four major milestones, each of which delivered working functionality. The milestones, schedule dates and the project groups (clusters) involved are shown in Table 8.1. The measured variables in this analysis are schedule estimates and actuals.
Table 8.1: Milestone / Schedule dates

<table>
<thead>
<tr>
<th>Milestone</th>
<th>Schedule Date</th>
<th>Clusters Measured</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>15th Sept 1997</td>
<td>CDOS, Defect, Requirement, Server Eng, SNAP-ON, UI, Test</td>
</tr>
<tr>
<td>M2</td>
<td>17th Nov 1997</td>
<td>As above minus Requirements</td>
</tr>
<tr>
<td>M3</td>
<td>26th Jan 1998</td>
<td>As above minus Requirements</td>
</tr>
<tr>
<td>M4</td>
<td>14th May 1998</td>
<td>As above minus Requirements, Server Eng, plus Test Design.</td>
</tr>
<tr>
<td>M5</td>
<td>5th June 1998</td>
<td>Test Design, Test Execution.</td>
</tr>
</tbody>
</table>

The Control Diagrams produced for each cluster were updated to show information for each given milestone cumulatively i.e. as each milestone was completed its data were added to that cluster’s Control Diagram. The sequence of Control Diagrams was repeated for each cluster. The milestone specific Control Diagrams for each cluster, enabled process performance to be monitored and corrected at the start of the subsequent milestone. The cumulative Control Diagrams showed how the process corrections affected the processes at each project milestone.

This section provides an example of the process performance analysis for one of the project clusters, the User Interface Development Cluster (UI). Figures 8.4 & 8.5 shows the performance of the User Interface Development Cluster with respect to estimate accuracy over the four milestones.
Figure 8.4: Control Chart for UI Cluster Estimates

The task of the User Interface cluster was to complete the replacement of an existing text based configuration entry system with a Windows based user interface. This version was also aimed at providing increased data entry productivity. The bulk of the replacement work had been carried out in a previous release. In this release it was understood from the earlier quality function deployment analysis, that the user interface work would be to complete the remaining entry screens and to add further productivity enhancements.

As part of the productivity enhancements, the user interface object structure needed some re-engineering. The re-engineering proved to be difficult to estimate accurately. The original estimates were low but corrective actions were put in place at the start of the second milestone. The Control Diagram shows an overall estimating accuracy of 76%. The cluster estimates did improve in later milestones as the corrective actions started to take effect.

First impressions seemed to indicate that the processes being used by the group were a little out of control, as Figure 8.5 indicates many of the accuracy values lay outside the middle third of the zone between the control limits. Furthermore, a simple inspection shows there is a noticeable skewed distribution. These anomalies were
sufficient to trigger a further investigation. The results indicated it was not the processes at fault, but a certain amount of misapplication by the cluster group concerned. The root cause turned out to be a cluster group dynamics issue. A simple restructuring of the cluster group corrected the problem for subsequent milestones.

Once the second milestone point had been added to the Control Diagram, it was clear that the cluster was having some problems. They had underestimated the scope of the work and were heading in to trouble. A review of the issues determined that there was no way back and the re-engineering, then started, had to be completed.

At this point specialist contractors were added to the project in order to complete the re-engineering and to free up full time team members for completion of the productivity enhancements. In this case adding resources was beneficial as the problem had been identified at an early stage and the contractor's learning curve was minimal. This resulted in an improvement both in estimating accuracy and acceptable deliverables.

The cluster also came in on budget. This was achieved because of the approach taken in requirements capture and enhancement prioritisation. Quality function deployment was used to determine which enhancements were to be carried out and what priority each would have. The enhancements were given priority levels 1 to 3. Expectations were set that priority 1 and 2 enhancements would be delivered. Priority 3 items would be delivered if project progress were better than expected.

The cluster's original cost profile was estimated on the basis of delivering all items (1, 2 and 3). As it turned out, the cluster delivered all priority 1 & 2 items, though this was achieved at the original predicted costs of doing all items. However, this shows that even potential disasters can be avoided if the right project planning and control processes are in place.
When the final milestone was completed and the set of four steps for carrying out analysis of the Control Diagram and the sigma histogram were applied, the cluster passed them all, with the exception of step four. However, as can be determined from the sigma histogram, the failure was only by ½% i.e. the target for a normal distribution is 68% of values within one sigma.

Table 8.2 shows a summary of the process performance for all clusters in the case study.

**Table 8.2: Process Performance Analysis Summary**

<table>
<thead>
<tr>
<th>Cluster</th>
<th>% within 1 sigma</th>
<th>% within 2 sigma</th>
<th>% within 3 sigma</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Interface</td>
<td>67.5%</td>
<td>97.5%</td>
<td>100%</td>
</tr>
<tr>
<td>SNAP ON</td>
<td>80.2%</td>
<td>98.6%</td>
<td>99%</td>
</tr>
<tr>
<td>Requirement</td>
<td>92%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Server Engineering</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>CDOS</td>
<td>59%</td>
<td>70%</td>
<td>98%</td>
</tr>
<tr>
<td>Test Design</td>
<td>93%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Test Execution</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Inspection</td>
<td>87.4%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>Defect Repair</td>
<td>60%</td>
<td>87.4%</td>
<td>100%</td>
</tr>
</tbody>
</table>
The results for the user interface cluster estimates were 67% within one sigma, 97% within 2 sigma and 100% within 3 sigma. An end of project process performance summary for all project clusters is shown in Table 7.3. Remember as the project team improves its applications of the processes, the upper and lower control point ranges will be reduced in order to drive more efficient project process execution.

8.5 Process capability measurement methodology

Each process in the QFSD framework has a number of work products associated with it. These work products are the characteristics and deliverables, which are the tangible evidence that a process is being applied.

Table 8.3 gives an example of the work products. The work products are further categorized into three classes depending on the strength and sophistication of each work product. In general, this classification is based on the researcher's experience of when the work product came into existence on past projects executed in the department over a period of six years. Generally, but not in every case, the earlier the work products became a regular deliverable on past projects, the lower the class they were assigned.

Table 8.3 shows the Input / Output work products associated with the planning process with the associated class assigned to each work product. For each work product a completion level is determined for the project being assessed. This is determined as a percentage of all work products being delivered. The completion level is kept as objective as possible based on actual deliveries. For example, the completion level of the work product "Each requirement validated with customers" would be calculated as a proportion of those requirements validated out of the total requirements.

Other evaluations were more subjective, however, such as "market window identified" where it is necessary to rely on the experience of those carrying out the task. In general, the subjective element has not been found to be a problem as the experience of the managers of the team has enabled reasonable estimates to be made. For each process the capability level can then be assessed. The capability is measured for each process as it is implemented for each individual project. As the
work products are delivered, the capability level of the process is increased. The
capability level of each process on completion of the project can then be used to
determine overall capability of the processes employed by the project team. This can
then be used to determine which processes need further improvement and
development.

The capability level for each process is determined as follows:

**Level 0:** A substantial number of 'objectives' as defined in the QFSD core model
have not been met. Notice that level 0 is assumed. If after initial inspection the
process is below level 0, then the full analysis is not worth applying, except for audit
purposes.

**Level 1:** All QFSD framework processes must be in place, unless specifically
highlighted as not applicable in the project management plan. A substantial number
of the 'objectives' must have been achieved. Seventy five percent of the Class 1
work products need to exist for applicable processes. In addition, the average %
score of each work product is equal to, or greater than 75%.

**Level 2:** All QFSD framework processes must be in place, unless specifically
highlighted as not applicable. All 'objectives', as defined in the QFSD implementation
model, have been achieved. Ninety percent of the level 1 work products and eighty
percent of the level 2 work products need to exist for applicable processes. In
addition, the average percentage scores for each work product at level 1 and 2, need
to be equal to, or greater than 90% and 80% respectively.

**Level 3:** All QFSD framework processes must be in place, unless specifically
highlighted as not applicable. All 'objectives', as defined in the QFSD implementation
model, have been achieved. Ninety percent of both level 1 & 2 work products and
eighty five percent of the level 3 work products, need to exist for applicable
processes. In addition, the average percentage scores for each work product at level
1, 2 and 3, need to be equal to, or greater than 90%, 90% and 85% respectively.

Work products or whole processes can be excluded from the capability analysis if they are deemed not applicable to a given project and are documented as such in the
project management plan. For example, "Concept Definition" was not required in this case study.

All capability analysis is carried out using a predefined capability analysis database with the required calculations and graphs generated within it. This spreadsheet is broken down by major QFSD process area (organisational, management, support, execution), then by individual sub process. Input and output work products with their class categories and completion levels are then logged for each sub process. The capability level can then be evaluated for each process.
Table 8.3: Planning work product analysis report

**Project Planning Process**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Work Products</th>
<th>Class</th>
<th>Completion %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Case</td>
<td>Type of contribution to the business documented.</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Market window identified</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Incremental supporting goals identified</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Required funds available</td>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Required resources available</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Are the business improvements quantifiable</td>
<td>3</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>Alternative approaches considered and documented</td>
<td>3</td>
<td>50</td>
</tr>
<tr>
<td>Project Goals</td>
<td>Goals defined in the Project Plan</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Systematic method for generation and review</td>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>Are all goal achievement constraints documented</td>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>Are the goals used to monitor project success</td>
<td>3</td>
<td>70</td>
</tr>
<tr>
<td>Requirements</td>
<td>Each requirement identified</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Each requirement validated with customers</td>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Each requirement validated with project goals</td>
<td>3</td>
<td>70</td>
</tr>
<tr>
<td>Outputs</td>
<td>Schedule</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maintainable schedule plan with delivery dates</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Schedule contains resource loading</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Milestones clearly identified and managed</td>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>Schedule contains actual vs planned dates</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Baseline schedule available</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Regular progress updates to schedule</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Critical path identified on schedule</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Maximum use of task concurrency shown</td>
<td>3</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>Schedule has realistic contingency / variance built in</td>
<td>3</td>
<td>45</td>
</tr>
<tr>
<td>Task Breakdown</td>
<td>Breakdown of project tasks</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Milestone mapped against tasks</td>
<td>2</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>Tasks given a unique identification code</td>
<td>2</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>Each task mapped against a resource</td>
<td>2</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>Each task given an individual completion date</td>
<td>3</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>Tasks given a priority order for execution</td>
<td>3</td>
<td>50</td>
</tr>
<tr>
<td>Estimates</td>
<td>Overall project estimate exists</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>Overall project cost exits (estimated / actual)</td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>Actual vs estimated estimates recorded</td>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>Estimates are recorded by individual task</td>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>Estimates are recorded by Dsg/Code/Test</td>
<td>3</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>Estimates are baselined during &amp; at the end of a project</td>
<td>3</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>Estimate are in line with historic project data</td>
<td>3</td>
<td>33</td>
</tr>
</tbody>
</table>

8.6 Example of the process capability analysis

The following provides an example of the capability analysis taken from the Control Desktop P2.0 project. For brevity, the example only covers project planning, in reality the analysis was applied to all project processes. In Table 8.6 the work products are shown for the QFSD frameworks 'Project Planning' process. As can be seen in the
table the work products are identified as inputs to and outputs from, the project planning process. The inputs and outputs are further grouped into specific project planning work tasks. Each task has a number of work products associated with it at each of the three class categories. A percentage score is used to identify the completion of each task and this was updated as the project proceeded.

A bar chart showing the completion levels achieved for each task for each class of work product was automatically generated from the database. The overall percentage completion level for each class of work product was also calculated and from this the process capability was then calculated.

These results were then balanced against a review of the ‘Project Planning’ objectives, as defined in the QFSD core framework, which is “To plan a software project which has an achievable schedule, and meets the company’s objectives”, i.e. the success of the project is reviewed in terms of schedule, costs and customer acceptance. If the capability analysis results in high levels of capability, the objectives will have a high probability of being achieved.

Figure 8.6: Bar Chart showing the completion levels for the project planning process for each work product class
Table 8.4 Capability analysis results for the Control Desktop p2.0 project

<table>
<thead>
<tr>
<th>Organisational Processes</th>
<th>Capability Level</th>
<th>Management Processes</th>
<th>Capability Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company Objectives</td>
<td>Level 3</td>
<td>Project Management Plan</td>
<td>Level 3</td>
</tr>
<tr>
<td>Business Case</td>
<td>Level 3</td>
<td>Project Planning</td>
<td>Level 3</td>
</tr>
<tr>
<td>Project Objectives</td>
<td>Level 3</td>
<td>Release Strategy</td>
<td>Level 3</td>
</tr>
<tr>
<td>Project Approval Request</td>
<td>Level 3</td>
<td>Resource Strategy</td>
<td>Level 2</td>
</tr>
<tr>
<td>Requirements Definition</td>
<td>Level 3</td>
<td>Requirements Capture</td>
<td>Level 3</td>
</tr>
<tr>
<td>Executive Test Plan Review</td>
<td>Level 3</td>
<td>Development Process</td>
<td>Level 2</td>
</tr>
<tr>
<td>Executive Release Readiness</td>
<td>Level 3</td>
<td>Risk Abatement</td>
<td>Level 2</td>
</tr>
<tr>
<td>Process Improvements</td>
<td>Level 3</td>
<td>Monitoring &amp; Control</td>
<td>Level 3</td>
</tr>
<tr>
<td>Support Processes</td>
<td>Capability Level</td>
<td>Execution Processes</td>
<td>Capability Level</td>
</tr>
<tr>
<td>Configuration Management</td>
<td>Level 3</td>
<td>Concept Definition</td>
<td>Level 3</td>
</tr>
<tr>
<td>Document Management</td>
<td>Level 2</td>
<td>Requirements Definition</td>
<td>Level 2</td>
</tr>
<tr>
<td>Quality Assurance</td>
<td>Level 2</td>
<td>Software Design</td>
<td>Level 2</td>
</tr>
<tr>
<td>Verification (inspection)</td>
<td>Level 3</td>
<td>Software Construction</td>
<td>Level 2</td>
</tr>
<tr>
<td>Validation</td>
<td>Level 3</td>
<td>Defect Repair Strategy</td>
<td>Level 2</td>
</tr>
<tr>
<td>Third Party Software &amp; Basis</td>
<td>Level 1</td>
<td>System Integration</td>
<td>Level 2</td>
</tr>
<tr>
<td>Defect Management</td>
<td>Level 2</td>
<td>Release Testing</td>
<td>Level 3</td>
</tr>
<tr>
<td>Process Improvement</td>
<td>Level 3</td>
<td>Organisational Readiness</td>
<td>Level 2</td>
</tr>
<tr>
<td>Change Control</td>
<td>Level 2</td>
<td>Software Maintenance</td>
<td>Level 1</td>
</tr>
</tbody>
</table>

The bar chart shown in Figure 8.6 is taken from the Control Desktop P2.0 project capability analysis and shows that the planning processes in the project were at level 3 with the overall scores for class 1 work products at level 1 being 98.6%, at class 2 being 99.2% and at class 3 being 89.98%.

Table 8.4 shows the levels achieved for all the processes in the QFSD framework. The process capability metrics give a high capability level for the majority of the processes used in the case study. Those processes, which resulted in a capability level less than level 3, were used as the focus for the process improvement goals for the next project. In particular, it has identified processes where questions needed to be raised about the relevance of the process, whether the techniques used were effective and whether the tools were adequate. Development effort could then be put into improving the process for the next project.
8.7 Conclusions

According to Weitz (1989), many software metrics programmes fail to live up to their potential because of a lack of commitment from project managers, team members or both. The recording and analysis of the metrics is seen as a time wasting diversion from the real task of developing the software. This chapter, however, has shown that within a well managed, quality development environment the collection of data need not be intrusive as there are clear benefits to be obtained in analysing the data that will be produced as part of the normal project management process.

This chapter has described the process performance model used at Fisher-Rosemount's U.K Technology Group at Leicester, UK. This has involved the processing of data commonly available within software project management. The results have then been presented for one part of one project, the User Interface development in the Control Desktop P2.0 project. It was shown that the information gathered and analysed very quickly drew attention to problems in the estimating and development processes. The speed in which the problems were then identified and remedial action taken enabled the project to get back on course and avoid the disastrous schedule and budget overruns was a significant improvement over Fisher-Rosemount Case Study One. The data analysis from the user interface cluster of the Control Desktop P2.0 project was clearly beneficial. This benefit was not unique to this one project cluster and similar results were obtained from other clusters, and indeed, in the majority of projects that have subsequently used this technique.

This chapter has also described a process for assigning and applying simple capability metrics to the software development processes employed. This helped identify which processes were in need of further development, allowing the developers to concentrate their efforts on improving these processes. As a result of the metrics programme, at the time of its completion, the Control Desktop P2.0 project was the most successful project executed by the Leicester Technology Team. The project was delivered within the best case release window, within budget and with high customer satisfaction. Given that the QFSD process and capability performance methodology has been applied to a number of projects at Fisher-
Rosemount with improving performance, this is a good indicator that achieving high capability is a learning process with the team improving with each new project. The benefits have been better managed, more cost and time effective projects producing higher quality software. Management and the development team have obtained a better understanding of the software development process and this continuing learning process has lead to a continuous improvement in both the development methodology and the resulting software.

This case study extension has shown here are significant benefits to be obtained from measuring process performance and capability. Yet the performance data used was commonly available data and the capability measurement methodology used was not complex or difficult. There are several well known process improvement and capability maturity measurement methodologies, reviewed by Hoyle (1998) or Primatesa (1994), but each involves a substantial level of commitment to be implemented. This case study has shown that, providing the software development processes can be well defined in terms of its sub processes and work products, it is not difficult to measure and then improve the capability of the processes. This simple methodology can then become a foundation for implementing one of the full process improvement or capability maturity methodologies at a later stage.

It is, therefore, concluded that the case study has shown that the key attributes of a successful performance and capability methodology have been achieved, as defined in the introduction, showing that clear benefits can be obtained from data that is commonly available in a well managed development environment.
Chapter 9 - Adaptability & Coverage of the QFSD Framework

The next step in the evolution of the QFSD framework was to take it out of an IT environment that had been traditionally used to working with quality standards and introduce it to an environment with no such history of adherence.

This opportunity arose when the researcher was appointed IT Director for the recently merged companies of AAH Pharmaceuticals and Lloyds Pharmacy. The merged company had a combined product portfolio of 98 software applications and had little or no experience in terms of quality standards, software engineering or project management. This chapter will show how the QFSD framework was adapted and extended in order to meet the challenges of the new more commercial environment.

Figure 9.1 shows the components of the QFSD framework that are extended and validated in a more commercial development environment this chapter.

![Diagram of the QFSD framework components](image)
9.1 Introduction

The new company not only had very poor software quality processes, for example there was no source control applied in development and no requirements documentation. In fact, product development groups seemed to discuss the requirements verbally with the business users, code and release software with no form of verification or validation. The problem was further compounded by the loss of key members of the IT teams as a result of the merger and the fact that the two companies had duplicate solutions for the same business requirement.

Introduction of the framework as it existed at that point would be a step too far for the new organisation, never mind extending the framework to cover more industry standard methodologies. Therefore, a four phase approach was planned as follows:

- **Phase one**: Introduction of emergency processes in order to stabilise the new IT organisation and maintain the current level of service to the business.
- **Phase two**: Establish quality framework requirements together with the new IT organisation and introduce the basic QFSD framework.
- **Phase three**: Introduce industry standard methodologies and extend the QFSD framework coverage.
- **Phase four**: Extend the QFSD framework in order to support major business change programmes, both locally and across multiple sites.

Given that the new organisation presented a green field site, albeit with a number of very poor practices and deeply ingrained resistance to change, each phase would take around twelve months in order to introduce and be able to see discernable improvement.

Note that the first two phases are considered in this chapter in terms of adaptability of the QFSD framework in a different IT environment and its coverage in terms of required processes. The second two phases are considered in the following chapter in terms of scalability of the QFSD framework by looking at the integration with commercial methods such as The Stationary Office (2002) and RUP (2007) and its use in a European Programme Management environment.
9.2 Phase one: Introduction of Emergency Processes

While the QFSD was not initially designed to be used for such emergencies, it did nevertheless prove to be an invaluable assistance in this phase, thus showing the flexibility of QFSD. Based on the recommendations of Matthew (2006), that indicated which of the QFSD processes to put in place first a "stripped down" version of the framework was quickly implemented.

During the first year the following fundamental processes were introduced to establish a solid foundation on which the QFSD Framework could be introduced so that the IT department could deliver the business change projects that would be required in subsequent years. Being a pharmaceutical wholesaler and retailer, these business change requirements would be driven by the NHS programme for health and related government changes in the way healthcare is delivered within the United Kingdom.

In terms of project management, a stripped down version of the QFSD framework management processes was introduced, which had a beneficial impact on the delivery of current projects and the planning of new projects. This included the following:

- The number of development environments was reduced to just three and a rule introduced that any project involving software development could only use a single development environment. This had the effect that developers had access to shared repositories and source code control systems in a consistent way and enabled integration of automated tools, such as those for software builds.

- Requirements Management was made mandatory in all projects involving software, whether that software was to be developed or purchased. A standard template for requirements capture was introduced from the QFSD framework.

- A basic design modelling approach was introduced for all projects of any significant size. This included modelling requirements as use cases and textual statements showing dependencies between them and the creation of logical,
physical, and, where appropriate, data models in the project analysis and
design phases.

- If appropriate, performance and capacity models would be created to document
how non-functional requirements were to be met.

- Testing and Defect Tracking processes were introduced from the QFSD
framework to manage and track outcomes of tests required too assess fully the
quality of an application or system.

- The CVS (2006) repository tool was introduced, together with a source code
control process in order to put all products under source code control.

- The change control process from the QFSD framework was introduced in order
to control all requirements, design and development change requests.

- Part of the QFSD online repository was used also as a basic project document
management repository, giving all projects an online project folder.

At the end of the first year an analysis of the major projects started and delivered in
that period was made in order to measure improvement. However, it was difficult to
get a like-for-like comparison with previous projects as no such quantitative
information had been captured. However, a number of key indicators were observed:

- A number of projects that had been continuously slipping were delivered.
  Notably a new dispensing application, which was the core business revenue
generating application of the retail business

- The endless backlog of projects had been rationalised with the business and a
  set of achievable projects agreed

- Improved predictability of project delivery, which can be seen in Table 8.1
  which compares favourably with the previous year when only four projects out
  of 27 running projects had been delivered

- A change control process had been introduced but more importantly it had
  been accepted by the business sponsors. This stopped the daily business
user’s visits to the developers’ desks, which had previously contributed to the non-delivery of projects.

Table 9.1 Phase one project delivery analysis

<table>
<thead>
<tr>
<th>Project Type</th>
<th>No of Lines of Code Changed</th>
<th>Schedule Success</th>
<th>Budget Success</th>
<th>Total Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Super market chain management application (new product)</td>
<td>32,500 lines</td>
<td>Planned success duration 6 months. Actual 8 months.</td>
<td>21% Overspend.</td>
<td>48 Man Months.</td>
</tr>
<tr>
<td>Hospital ordering application (new product)</td>
<td>26,600 lines</td>
<td>Planned success duration 8 months. Actual 10 months.</td>
<td>15% Overspend.</td>
<td>34 Man Months.</td>
</tr>
<tr>
<td>New splitting module for logistics (replacement module)</td>
<td>1,700 lines</td>
<td>Planned success duration 2 months. Actual 2 months.</td>
<td>On budget.</td>
<td>5 Man Months.</td>
</tr>
<tr>
<td>Refit of warehouse in Romford (installation and configuration)</td>
<td>1,100 lines</td>
<td>Planned success duration 2 months. Actual 3 months.</td>
<td>27% Overspend.</td>
<td>6.5 Man Months</td>
</tr>
<tr>
<td>Refit of warehouse in Leeds (installation and configuration)</td>
<td>1,200 lines</td>
<td>Planned success duration 3 months. Actual 3.5 months.</td>
<td>12% Overspend.</td>
<td>6.2 Man Months</td>
</tr>
<tr>
<td>Electronic van delivery (new development)</td>
<td>5,600 lines</td>
<td>Planned success duration 7 months. Actual 10 months.</td>
<td>23% Overspend.</td>
<td>28 Man Months</td>
</tr>
<tr>
<td>New windows based dispensing application (new product, only)</td>
<td>2,700 lines</td>
<td>Planned success duration 6 months. Actual 9</td>
<td>On Budget.</td>
<td>19.5 Man Months</td>
</tr>
</tbody>
</table>
Table 9.1 only shows a subset of the larger projects, but the improvement trend was evident across the majority of projects. The IT department comprised of 250 staff, 90 of which were software developers. In a twelve month period the department would complete about 130 projects comprising of software developments, enhancements, change requests, hardware and network deployments. However, Table 9.1 is a random sample of the projects carried out following the phase one changes and whilst projects were overrunning and often outside budget, they were being delivered and the business users were seeing the value from their IT investments. In terms of the QFSD framework the results were incorporated in to the basic building blocks of the QFSD by identifying those processes that are the fundamental first adoption steps. These steps were included in the guidance documents associated with the QFSD.

9.3 Phase two: Establish quality framework requirements together with the new IT organisation and introduce the basic QFSD framework.

9.3.1 Establish quality framework requirements

The requirements for a quality software framework within a commercial environment in which projects include small modifications to exiting applications, creation of new minor applications and major new applications for deployment on multiple sites whilst being connected back to a central head office, was quite different from the software product development environment in which the QFSD framework had been initially evolved and validated. Therefore, rather than simply introducing the existing 'core' framework an extensive review of the quality processes required within this new environment was carried out with the IT departmental managers in order to establish the fit between the framework and the actual needs of the departments.
The process followed was:

1. Identify and involve all staff who were associated with the planning, execution and delivery of projects within the department in order to understand and document the lifecycle of each type of project e.g. small change request, defect solution, minor new development, major new development and reengineering.

2. Capture feedback from the phase one process changes and any additional process requirements and compare them with the main QFSD framework. This was achieved by holding workshops with the project managers, development groups and project stakeholders. Changes proposed were then reviewed by the senior project, development and stakeholder management and raised as change requests against the framework.

3. Carry out an exercise to map all project roles, responsibilities and known or identified processes on to the project lifecycles as defined in 1 above. Again this was carried out by the senior project, development and stakeholder management.

4. Split the project management and development process out and define the communication interfaces between them as these two areas were generally managed by separate disciplines.

5. Identify future methodologies with which the QFSD framework would need to be integrated. In order to establish these future methodologies a review that involved the pharmaceutical suppliers and governmental healthcare agencies was carried out. The results indicated that in order to continue to do business, Celesio AG would be required to adopt industry standard methodologies. Whilst the standards varied across each of the 16 European countries in which Celesio AG does business, they were distilled down to two fundamental methodologies. The first is a recognised project management methodology and the second a development methodology, which included the whole development, test and release lifecycle.
6. Compare the 'required' lifecycles and processes with those contained in the QFSD framework

The results from the review were as follows:

- The QFSD framework 'Core' processes and associated online repository provided a good fit with the process requirements established during the review. However, a level of customisation of the processes would be carried out in order to fit them exactly to the established requirements. In addition, it was identified that an additional top-level quality process document would be required, which should be adopted by the QFSD framework and should act as the company's top-level software development process standard. The document was called the Product Development Procedure (O'Neill, 2002). This document should also define the agreed software development lifecycle model. The definition of the lifecycle model for Celesio is described in Table 9.2.

- The interfaces between the project management and development processes required matched those within the QFSD framework.

- The requirement for integration of industry standard methodologies could not be fully provided at this point, but it was agreed to evaluate the integration of the The Stationary Office (2002) project management methodology and the RUP (Kruchten, 1999) development methodology. The approach was to first introduce both methodologies as pilots and then integrate them in to the QFSD framework.

9.3.2 New IT organisation

At the end of the first year it became necessary to change the IT organisation to better fit the needs of both the business and to enable the QFSD framework process to be more efficiently executed. Whilst this topic is not described in detail it should be born in mind that for processes to be optimised the organisation that uses them will similarly need to be optimised.
During phase one, an extensive recruiting campaign had secured for the company an additional twenty five software developers, four software system architects and four experienced software project managers, plus two staff experienced in running a project office.

Rather than having a large pool of software engineering that managers previously had to complete for in order to run their projects, plus a separate operations and support group, the department was split into the following groups:

- Four software development groups, which were aligned with the main business customers: Wholesale Logistics, Wholesale Marketing, Retail Operations and Retail Pharmacy. Each group had its own dedicated software architect, software developers and a senior project manager. Each software development group was managed by a senior technical manager. This organisation was fully supported by the businesses as they could see a dedicated group of resources assigned to their projects and helping to meet their business objectives.

- Each of the project managers reported into the project management group, which was managed by the head of projects. The head of projects was also responsible for the project office (structure and functions as defined by CCTA (1999)), which in turn was responsible for the central QFSD repository, audit of compliance to the QFSD processes in terms of process performance and capability and for training of new staff in the adoption of the QFSD processes.

- A separate infrastructure design and implementation group was established in order to build-up a centre of excellence for infrastructure in order that infrastructure modernisation could be clearly separated from projects.

- A separate operations and support group was also put in place in order to provide improved pre- and post-deployment support and to enable a focus on quality improvement processes to be made which were specific to operations e.g. The IT Infrastructure Library ® (2008). The integration of Infrastructure Library ® into the framework will be considered as a future enhancement to the QFSD framework.
The new organisation was designed to give the various business sponsors the view that they had dedicated resources and hence give them confidence that their needs were being addressed. Further, the new organisation would enable the introduction, management and support of the new QFSD processes, by providing clear responsibility assignment for each of the major framework process areas as follows:

- **Organisational processes:** Responsibility assigned to the project management department.

- **Management processes:** Responsibilities have a defined split between the project management and development departments.

- **Implementation processes:** Responsibility assigned to the development and Infrastructure departments.

- **Support processes:** Responsibility assigned to the operations and support department.

### 9.3.3 Introduction of the basic QFSD framework

As part of phase one, a substantial number of the ‘core’ QFSD framework processes had already been introduced during the emergency process introduction activity with good success. This success was encouraging and enabled the senior management team in the company to be convinced to approve the next phase. The first step was to revise the IT department’s structure as described in the previous section in order to delivery the benefits of the new framework efficiently.

At this point, the foundation for the introduction of the full ‘core’ QFSD framework was in place. However, whilst the QFSD framework could be introduced in a similar fashion to that at Fisher-Rosemount, this new environment was much more complex, with many different types of projects and business sponsors. The fit between the current assumed lifecycle model of the ‘core’ QFSD framework and the requirements of this new environment need to be evaluated and adjusted if necessary.

In addition, the overall lifecycle model within which the QFSD processes were to be used needed to be tuned to the types of project and, during this tuning process, there
was a need to gain the buy-in of all staff responsible for project deliveries. It should be understood that the views of software development from a project manager's perspective is very different to those of a software development manager, but both must accept a common lifecycle model in order to deliver quality software (Davis et al., 1998).

Therefore the main purposes of designing and documenting the overall project lifecycle process for each project category are to:

- Enable all persons concerned with creating, planning and executing projects to understand the process to be followed during the life of the project.
- Capture the best experience within the organization so that the lifecycle process can be improved continually and duplicated on future projects.
- Enable all project roles, responsibilities, project planning, estimating, scheduling, monitoring, control methods and tools, to be appropriately related to the overall QFSD lifecycle processes: Organisational, Management, Supporting and Implementation.

Unless a well-documented, easily understandable picture of the lifecycle processes exists it is difficult to achieve the full benefits.

At this point a further requirement, which is discussed in Chapter 9 in detail, was also taken in to account. This requirement was to enable the lifecycle model to integrate with industry standard methodologies. On the face of it this looks like an impossible requirement as the number of methodologies for project management and software development is significant. However, the researcher took a pragmatic view on this by selecting the PRINCE2 methodology (waterfall) for project management and the RUP methodology (iterative) for software development. Based on the literature search carried out for this thesis these two methodologies have core principles that are reflected in the majority of modern project management and software development methodologies. Therefore, by integrating these two representative methodologies this should provide the foundations for integrating a significant number of similar methodologies.
The actual lifecycle model was to be created by the involvement of all senior IT staff (head of software development, head of business analysis, head of operations, head of project management), senior business sponsors (logistics director, pharmacy director and marketing director) and documented in a single top-level standard known as the Product Development Procedure (PDP). A working party was created and chaired by the researcher in order to develop the new lifecycle model.

9.3.3.1 Defining the lifecycle model

In designing and documenting the lifecycle model the working party considered the following areas:

- Identification of the main development phases and sub-phase within each project.
- Which of the phases could be executed in sequence and which could be executed in parallel? This also applied to the identified sub-phases.
- Which phases can be repeated and how all phases and sub-phases are inter-related.
- The number and placement of decision gates such as business case, investment appraisal, major development milestones etc.

Once the above was established the working party identified the delivery artefacts to be produced from each phase. These artefacts, for the most part, existed as part of the ‘core’ QFSD framework:

- Documents & tools related to the organisational and management processes identified:
  - Business case
  - Project approval request
  - Project management plan
  - Schedules (Gantt, Work Breakdown Activities, etc)
  - Resource estimate template
• Cost template
• Risk management plan and risk log
• Change notices template
• Quality standards documentation
• Progress report templates
• Release planning template
• Phase release approval template

• Documents and tools relating to supporting processes identified:
  • QFSD online repository: holds all project related documentation
  • Defect management procedure and tool
  • Project management planning tool
  • Risk management tool
  • Quality standards related to software development, testing and support
  • Change request procedure and tool
  • Automated regression testing procedure and tool
  • Configuration management procedure
  • Verification (inspection procedure and tool)
  • Third party software reuse procedure
  • Process performance procedure
  • Process improvement procedure

• Documents related to the Implementation processes identified:
  • Requirements specification templates
  • Requirements process modelling templates and tools
  • Design specification templates
• Design modelling templates and tools
• Detailed design specifications template
• Software architecture templates
• Infrastructure deployment templates
• Development methodology procedure (cauterisation in this case)
• Defect repair procedure
• Test specification templates
• Test case templates
• User acceptance templates
• Test results templates
• System integration procedure
• Operator templates
• Maintenance and diagnostics templates
• Maintenance procedure
• User documentation templates
• Architecture verification templates
• Pilot testing templates

The next step was to further define the decision points, which occur at the start and the end of each phase or sub-phase. The decision points typically found were the following:

• To approve funds to be expended
• To continue with planned activities within the current phase
• To start work on the subsequent phase
• To re-plan and re-execute a completed phase
o To revise project objectives, plans and schedule when major project changes are to be made

o To terminate the project when it is clear that the objectives or return on investment targets cannot be met

9.3.3.2 Documenting the lifecycle model

The lifecycle model was then documented in a standards document known as the Product Development Procedure (PDP) and the relevant modifications made to the QFSD online repository application. The PDP was constructed such that it could be easily translated to an online document and integrated into the existing QFSD online repository.

The QFSD 'core' framework provides its own lifecycle model: processes, templates and tools, part of which the department had already been using during the first phase (phase one). However, part of the adoption of QFSD into a new environment requires the lifecycle model to be tailored in order to provide an optimised software development capability. The requirements for the tailoring were generated by senior IT and business staff as described in the previous section. The requirements were focused in two areas. The first being the definition of an overall lifecycle phase model, which is described in Table 9.2 and shown graphically in Figure 9.2.

Table 9.2 Overall lifecycle phase model

<table>
<thead>
<tr>
<th>Lifecycle Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>0 – Start Up</strong></td>
<td>The start up task includes all activities prior to the project being formally approved. Essentially all activities associated with the production and approval of the project approval request document (PAR), overall requirements and the business case.</td>
</tr>
<tr>
<td><strong>A – Initiation &amp; Planning</strong></td>
<td>All activities associated with the production of the initial Project Management Plan (PMP), initial project planning, estimates, risk assessment and process</td>
</tr>
<tr>
<td>Lifecycle Phase</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>B – Requirements Capture</strong></td>
<td>All activities associated with the capture and detailed recording of the statement of requirements.</td>
</tr>
<tr>
<td><strong>C – Functional Specification &amp; High Level Design</strong></td>
<td>All activities associated with the production of the technical specification and top level design.</td>
</tr>
<tr>
<td><strong>D – Investment Appraisal</strong></td>
<td>At this point, the project will have sufficient planning, design and scope information to make a realistic investment appraisal of the project. This is one of the major project management framework stage gates at which the project could be re-submitted for approval.</td>
</tr>
<tr>
<td><strong>E – Detailed Design, Code &amp; construction</strong></td>
<td>All activities associated with detailed design, code, defect management, unit testing and release test strategy.</td>
</tr>
<tr>
<td><strong>F – Test</strong></td>
<td>All activities associated with unit, system, integration and regression testing and user acceptance testing.</td>
</tr>
<tr>
<td><strong>G – Implementation</strong></td>
<td>All activities associated with delivery of the system, pilots, maintenance, support and end user training.</td>
</tr>
<tr>
<td><strong>H – Close Down</strong></td>
<td>All activities associated with shutting down a project. Closing down a project is a rolling process. Once the software has been delivered to the customer, the project holds a project post-mortem to review performance of the project execution. From these meetings, recommendations for process improvements are made. Further, down the line input from the customer is taken to establish how successful the project has been. Statistics such as number of crashes, defects reported performance are collected and released.</td>
</tr>
<tr>
<td>Lifecycle Phase</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>X - All phases</td>
<td>These activities can be associated with all phases of a project and cover 'support' processes as defined in the 'core' QFSD framework.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Repeated per project cluster or increment</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0)</td>
</tr>
<tr>
<td>(A -&gt; B)</td>
</tr>
<tr>
<td>(C -&gt; E)</td>
</tr>
<tr>
<td>(D -&gt; F -&gt; G)</td>
</tr>
<tr>
<td>(H)</td>
</tr>
<tr>
<td>(X)</td>
</tr>
</tbody>
</table>

Figure 9.2 Lifecycle model

The second focus for requirements was a list of artefact deliverables as defined in the previous section. Both the lifecycle phases and the artefact deliverables were then mapped on to the QFSD 'core' framework. The results of this analysis are shown in Appendix F. The table in this appendix groups the QFSD process areas and processes against the new lifecycle phases and the required deliverables.

The mapping of the QFSD processes, although shown at a high level in the table, was the result of a careful cross mapping of the QFSD processes and existing guidelines, templates and tools against the new lifecycle delivery artefact requirements. It is worth remembering that the QFSD 'core' model (refer to Appendix
C) contains online guidelines, templates and tools for each process. As a result of this mapping process only two additional processes needed to be added to the QFSD framework, together with a number of changes to the existing delivery artefact templates. The main change was to the top-level navigation flow in the QFSD online repository to use the new lifecycle phase model and the addition of the Product Development Plan document itself to the repository.

9.3.3.3 Analysis of phase two improvements

At the end of the second year an analysis of the major projects started and delivered in that period was again made in order to measure improvements due to the introduction of the tailored QFSD framework and revised lifecycle model. Unlike the analysis at the end of the first year application of the QFSD framework enabled additional metrics concerning performance and capability to be measured using the methods and tools provided within the framework. Therefore the project analysis was carried out using the following three metrics types with the audit and collation of the metrics now being carried out by the new project office:

1. Performance against budget and schedule

2. Process performance using QFSD control diagram approach (refer to Chapter 8)

3. Process capability using QFSD process capability data repository (refer to Chapter 8)

The first analysis, performance against budget, which provides a direct comparison with the analysis results generated at the end of phase one involved the capture of basic budgetary and schedule performance metrics. This information was now automatically available from the project office that now had direct access to the QFSD central repository containing each project's Project Management Plan, Detailed Schedule and associated timesheet recording and Project Milestone Tracking information.

The key improvement indicators that were observed during phase two were:
• Projects were now following an agreed methodology and lifecycle, with QFSD processes being introduced and monitored by the project office.

• On average a 12% improvement in estimating and 9% improvement in schedule predictability was achieved across all projects. However, the Table 8.3 shows a subset of all projects used to generate the statistics and hence shows 9% and 6.5% respectively for the sub-set of projects considered in the table.

• The number of projects being executed during the year was also reduced from 130 to 84. On investigation this was due to projects scope being properly reviewed with the business sponsors, such that on average project scope was increased by 22%, which is a result of including related enhancements in to a single project, rather than having to create smaller projects for “things we forgot” later in the year.

• There was an improved relationship with the various business sponsors with improved customer survey results in terms of meeting business objectives and business case ROI targets, which the project office monitored on behalf of the businesses.
Table 9.3 Phase two project delivery analysis for selected projects

<table>
<thead>
<tr>
<th>Project Type</th>
<th>No of Lines of Code Changed</th>
<th>Schedule Success</th>
<th>Budget Success</th>
<th>Total Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time management system (new development)</td>
<td>22,400 lines.</td>
<td>Planned duration 8 months. Actual 9.5 months.</td>
<td>12% Overspend.</td>
<td>54 Man Months.</td>
</tr>
<tr>
<td>Re-engineering of pricing application (complex)</td>
<td>14,600 lines.</td>
<td>Planned duration 6 months. Actual 7.2 months.</td>
<td>8.5% Overspend.</td>
<td>26 Man Months.</td>
</tr>
<tr>
<td>Ward order assembly (new development)</td>
<td>15,245 lines</td>
<td>Planned duration 5 months. Actual 5.3 months.</td>
<td>4.6% Overspend</td>
<td>24 Man Months.</td>
</tr>
<tr>
<td>Refit of warehouse in Southampton (installation and configuration)</td>
<td>1,400 lines</td>
<td>Planned duration 2 months. Actual 2 months.</td>
<td>3% Overspend</td>
<td>6 Man Months</td>
</tr>
<tr>
<td>Mediate replacement (new development)</td>
<td>12,460 lines</td>
<td>Planned duration 5 months. Actual 5.4 months.</td>
<td>7.2% Overspend.</td>
<td>28 Man Months</td>
</tr>
<tr>
<td>20 line invoice (enhancement)</td>
<td>230 lines</td>
<td>Planned duration months. 1 Actual 1 months.</td>
<td>0% Overspend.</td>
<td>2 Man Months</td>
</tr>
<tr>
<td>AAH Point version 2 (major enhancement)</td>
<td>840 lines</td>
<td>Planned duration 3 months. Actual 3.3 months.</td>
<td>On Budget.</td>
<td>16 Man Months</td>
</tr>
<tr>
<td>Pharmacy Point Europe (adoption of standard)</td>
<td>230 lines</td>
<td>Planned duration 4.5 months.</td>
<td>17% Overspend.</td>
<td>8 Man Months</td>
</tr>
</tbody>
</table>

Pharmacy Point Europe (mostly data configuration)
The second analysis, process performance using the QFSD control diagram approach, considered a number of basic process performance metrics, which it used in each of the projects in the previous table. However, given that the metrics were not captured during phase one, it was necessary to provide a comparison of metrics captured for a project on a cluster by cluster basis to see if the project was improving its execution efficiency by learning from the previous clusters analysis. Note, all projects in phase two use the QFSD clusterisation approach as described earlier in the thesis and defined in Appendix (C).

Comparison across projects would not be meaningful as the projects are for the most part very different.

For each of the projects in the previous table the first three development clusters were analysed using the QFSD process performance method using accuracy of the clusters development estimates, quality of the required artefacts delivered by the cluster and errors discovered in project documentation due to the introduction of the Fagan (1986) inspection technique.

Tables 9.4 and 9.5 show the results from the first project, executed in the first quarter of the year and the last project executed in the fourth quarter of the year. These tables show the dramatic difference in performance improvement between the projects using the full QFSD Framework when it was first introduced and those that were using it as the process began to mature a year later within the company.

Table 9.4 Phase two cluster process performance analysis - Time Management System

<table>
<thead>
<tr>
<th>Project: Time management system</th>
<th>Cluster</th>
<th>Process metric</th>
<th>% within 1 sigma</th>
<th>% within 2 sigma</th>
<th>% within 3 sigma</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>Accuracy of estimates</td>
<td>57.5 %</td>
<td>77.5 %</td>
<td>92.0 %</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Quality of artefacts</td>
<td>63.2 %</td>
<td>75.6 %</td>
<td>91.0 %</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Errors discovered</td>
<td>63.0 %</td>
<td>76.0 %</td>
<td>81.0 %</td>
</tr>
<tr>
<td>Cluster</td>
<td>Process metric</td>
<td>% within 1 sigma</td>
<td>% within 2 sigma</td>
<td>% within 3 sigma</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>----------------------</td>
<td>------------------</td>
<td>------------------</td>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Accuracy of estimates</td>
<td>62.0 %</td>
<td>79.0 %</td>
<td>96.0 %</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Quality of artefacts</td>
<td>65.4 %</td>
<td>78.6 %</td>
<td>94.5 %</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Errors discovered</td>
<td>67.0 %</td>
<td>78.0 %</td>
<td>94.0 %</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Accuracy of estimates</td>
<td>67.0 %</td>
<td>84.6 %</td>
<td>97.2 %</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Quality of artefacts</td>
<td>70.5 %</td>
<td>84.9 %</td>
<td>96.5 %</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Errors discovered</td>
<td>72.0 %</td>
<td>86.0 %</td>
<td>92.0 %</td>
<td></td>
</tr>
</tbody>
</table>

### Table 9.5 Phase two cluster process performance analysis – Pharmacy Point Europe

The two tables show that there is a continuous improvement in the metrics scores within a given project as the teams learn and benefit from the measurements and associated corrective actions applied by the project management. In addition, there is also a continuous improvement shown between the projects as the QFSD framework matures within the development teams. This continuous improvement was shown to be incremental throughout the year when the results from each of the eight projects were performance analysed using the same three metrics.

The third analysis, using the QFSD process capability data repository, considered all QFSD processes to measure the improving process capability of each project. The same two projects as used for the process performance analysis are shown here:

Note that in the Tables 9.6 and 9.7:

- Level 1 is considered basic capability in the QFSD process
- Level 2 is considered intermediate capability in the QFSD process
Level 3 is considered advanced capability in the QFSD process

Table 9.6 Phase two project process capability analysis – Time recording system

<table>
<thead>
<tr>
<th>Organisational Processes</th>
<th>Capability Level</th>
<th>Management Processes</th>
<th>Capability Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company Objectives</td>
<td>Level 1</td>
<td>Project Management Plan</td>
<td>Level 2</td>
</tr>
<tr>
<td>Business Case</td>
<td>Level 1</td>
<td>Project Planning</td>
<td>Level 1</td>
</tr>
<tr>
<td>Project Objectives</td>
<td>Level 2</td>
<td>Release Strategy</td>
<td>Level 2</td>
</tr>
<tr>
<td>Project Approval Request</td>
<td>Level 2</td>
<td>Resource Strategy</td>
<td>Level 2</td>
</tr>
<tr>
<td>Requirements Definition</td>
<td>Level 1</td>
<td>Requirements Capture</td>
<td>Level 1</td>
</tr>
<tr>
<td>Executive Test Plan Review</td>
<td>Level 2</td>
<td>Development Process</td>
<td>Level 2</td>
</tr>
<tr>
<td>Executive Release Readiness</td>
<td>Level 1</td>
<td>Risk Abatement</td>
<td>Level 1</td>
</tr>
<tr>
<td>Process Improvements</td>
<td>Level 3</td>
<td>Monitoring &amp; Control</td>
<td>Level 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Test Management</td>
<td>Level 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Problem Resolution</td>
<td>Level 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Support Processes</th>
<th>Capability Level</th>
<th>Execution Processes</th>
<th>Capability Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Management</td>
<td>Level 2</td>
<td>Concept Definition</td>
<td>Level 1</td>
</tr>
<tr>
<td>Document Management</td>
<td>Level 2</td>
<td>Requirements Definition</td>
<td>Level 2</td>
</tr>
<tr>
<td>Quality Assurance</td>
<td>Level 1</td>
<td>Software Design</td>
<td>Level 2</td>
</tr>
<tr>
<td>Verification (inspection)</td>
<td>Level 1</td>
<td>Software Construction</td>
<td>Level 2</td>
</tr>
<tr>
<td>Validation</td>
<td>Level 2</td>
<td>Defect Repair Strategy</td>
<td>Level 1</td>
</tr>
<tr>
<td>Third Party Software &amp; Reuse</td>
<td>Level 1</td>
<td>System Integration</td>
<td>Level 2</td>
</tr>
<tr>
<td>Defect Management</td>
<td>Level 2</td>
<td>Release Testing</td>
<td>Level 1</td>
</tr>
<tr>
<td>Process Improvement</td>
<td>Level 2</td>
<td>Organisational Readiness</td>
<td>Level 1</td>
</tr>
<tr>
<td>Change Control</td>
<td>Level 2</td>
<td>Software Maintenance</td>
<td>Level 1</td>
</tr>
</tbody>
</table>
Table 9.7 Phase two project process capability analysis – Pharmacy Point Europe

<table>
<thead>
<tr>
<th>Organisational Processes</th>
<th>Capability Level</th>
<th>Management Processes</th>
<th>Capability Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company Objectives</td>
<td>Level 2</td>
<td>Project Management Plan</td>
<td>Level 3</td>
</tr>
<tr>
<td>Business Case</td>
<td>Level 2</td>
<td>Project Planning</td>
<td>Level 2</td>
</tr>
<tr>
<td>Project Objectives</td>
<td>Level 2</td>
<td>Release Strategy</td>
<td>Level 2</td>
</tr>
<tr>
<td>Project Approval Request</td>
<td>Level 2</td>
<td>Resource Strategy</td>
<td>Level 2</td>
</tr>
<tr>
<td>Requirements Definition</td>
<td>Level 2</td>
<td>Requirements Capture</td>
<td>Level 3</td>
</tr>
<tr>
<td>Executive Test Plan Review</td>
<td>Level 2</td>
<td>Development Process</td>
<td>Level 2</td>
</tr>
<tr>
<td>Executive Release Readiness</td>
<td>Level 1</td>
<td>Risk Abatement</td>
<td>Level 3</td>
</tr>
<tr>
<td>Process Improvements</td>
<td>Level 3</td>
<td>Monitoring &amp; Control</td>
<td>Level 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Support Processes</th>
<th>Capability Level</th>
<th>Execution Processes</th>
<th>Capability Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Management</td>
<td>Level 2</td>
<td>Concept Definition</td>
<td>Level 1</td>
</tr>
<tr>
<td>Document Management</td>
<td>Level 3</td>
<td>Requirements Definition</td>
<td>Level 2</td>
</tr>
<tr>
<td>Quality Assurance</td>
<td>Level 2</td>
<td>Software Design</td>
<td>Level 2</td>
</tr>
<tr>
<td>Verification (inspection)</td>
<td>Level 3</td>
<td>Software Construction</td>
<td>Level 3</td>
</tr>
<tr>
<td>Validation</td>
<td>Level 2</td>
<td>Defect Repair Strategy</td>
<td>Level 2</td>
</tr>
<tr>
<td>Third Party Software &amp; Rare</td>
<td>Level 1</td>
<td>System Integration</td>
<td>Level 2</td>
</tr>
<tr>
<td>Defect Management</td>
<td>Level 2</td>
<td>Release Testing</td>
<td>Level 2</td>
</tr>
<tr>
<td>Process Improvement</td>
<td>Level 3</td>
<td>Organisational Readiness</td>
<td>Level 2</td>
</tr>
<tr>
<td>Change Control</td>
<td>Level 2</td>
<td>Software Maintenance</td>
<td>Level 1</td>
</tr>
</tbody>
</table>

From the process capability analysis it can be seen that the levels of capability of the project team improved considerably over the period of phase two. It should be remembered that the team involved in the first project at the start of phase two had experienced the basic 'emergency' QFSD processes during phase one, so in effect they had been exposed to the QFSD framework for approximately 12 months. The resulting capability analysis for the Time Recording Project would be comparable with a CMM (2004) level 2 organisation based on the CMM guidelines (Caputo, 1998).

By the end of phase two the Pharmacy Point Europe project team had reached a level 3 organisation based on the CMM guidelines Caputo, (1998). Whilst this level was not uniform across the whole of the development department, it was clear that the capability maturity level was improving rapidly. Areas that were showing a very high degree of maturity were:
- Document management due to the introduction of the QFSD repository and its enforcement by the project office
- Project management processes, which are very detailed and supported by the new Celesio lifecycle model
- Process improvements, again supported by the project office and benefiting from the process performance and capability methods and tools provided by the QFSD framework
- Requirements capture and prioritisation enabled by the adoption of the QFD method
- Project reviews benefiting from the adoption of the Fagan (1986) inspection method.

The maturity level continued to improve as the teams became familiar with the methodology and through the process improvement process began to further optimise the QFSD framework. The next step in the introduction was to enable industry standard methodologies to be integrated into the QFSD framework; this is addressed in the next phase.

9.4 Conclusions

The introduction of any quality methodology into an organisation is a difficult process in itself, even if the organisation is already predisposed to working with quality standards. However, the challenge presented to the adoption of the QFSD framework in Celesio was significantly more difficult. Celesio were not used to working to any recognised quality standards with the IT function operating with no discernable methodology. The challenge was also intensified as the new environment was much more commercial that that used to establish the original framework, with a greater number of projects being executed in parallel and much shorter project schedules.

The first two phases of the QFSD Framework introduction lasted two years and initially introduced the basic IT quality processes from the framework, which were:
- Document management in the form of the QFSD online repository
• Configuration management
• Requirement management
• Test and defect tracking
• Source code management
• Change control
• Rationalisation of the development environments
• Introduction of basic project management methodology

This was followed by the introduction of the full and tailored QFSD Framework. During the first phase there was a dramatic improvement in the rate of projects being delivered, albeit with an average 25% overspend and schedule delay. However, given that in previous years a significant number of projects either failed to deliver, or did not meet the business users expectations, this was a great improvement. Having effectively stabilised the IT environment and introduced a basic culture of software quality the next step was to introduce the full QFSD framework.

Based on the improvements generated during the first phase, the senior management team supported the full introduction of the QFSD framework. During the second phase, the QFSD Framework was fully introduced along with a strong Project Office and a suitable organisational restructure. There were a number of drivers for the restructure, but one of the main drivers was to have an organisation that would enable the efficient delivery of IT projects that met the business expectations.

Introduction of the framework required the tuning of the basic lifecycle model in order to better fit the new environment. In reality, the tuning was more about gaining the buy-in from two of the major departments, that of Project Management and Development. This coupled with a full review of the QFSD processes resulted in a high level of agreement between all IT departments, which cumulated in the publication of the Celesio Product Development Procedure (PDP).
At the end of phase 2 it was now possible to carry out a more quantitative analysis of the improvements. In summary the improvements observed and presented to the Celesio Senior Management Team were as follows:

- All projects now follow the standard as defined in the Celesio PDP and are much more predictable in terms of effort and timescale
- All project artefacts are managed in a central document repository (online QFSD repository), which enables the Project Office to apply governance and carry out audits successfully
- Based on the Process Performance and Process Capability analysis carried out by the Project Office the level of maturity of the Celesio IT organisation is above that of a typical IT organisation. The researcher's experience of having assisted a number of companies in achieving ISO accreditation shows that the level of maturity is equivalent to CMM (2004) level 2. However, in order to gain ISO 9003 TickIT (1994) the IT department would need to gain more experience and generate another year of using the framework before applying for accreditation.

This chapter has shown that the QFSD framework proved sufficiently flexible in handling a completely different environment to that at Fisher-Rosemount. In many respects Celesio AG was a 'green-field' site with little or no software processes in place. By the end of second phase of the QFSD introduction, the Celesio AG IT organisation had matured significantly in terms of software engineering processes and could be considered as a level 2 organisation based on the CMM guidelines (Caputo 1998).

Experience was gained in the adoption and tailoring of the QFSD framework to a specific organisation. It became clear that the fundamental lifecycle model associated with the QFSD framework should always be 'tuned' by the relevant IT management during the initial phase for a number of reasons. Firstly, not all IT environments are the same and hence the model needs to be optimised. Secondly, by involving the key IT management in the tuning process a level of buy-in and ownership is generated. Thirdly, once agreed, the life-cycle model is used as the
'backbone' upon which the QFSD processes are deployed. This 'backbone' was also used to generate the top-tier quality document known as the Product Development Procedure' (PDP).

A further result that came out of the need to carry out an emergency implementation of the framework was the identification of a fundamental list of QFSD processes that must be deployed as the first step in adoption. In addition, to this, the need to align the roles and responsibilities of the IT management in order that clear responsibilities for each QFSD process area and its associated deliverables was established.

Finally, it was also established during the first two phases that the QFSD framework must be sufficiently flexible to be integrated with other industry standard methodologies. This integration topic is discussed in Chapter 10.
10 Chapter 10 - Scalability of the QFSD Framework

This Chapter covers phase three of the introduction of the QFSD Framework into the commercial software development environment of Celesio. Phase three covers the introduction of industry standard methodologies, which also addresses two further QFSD design objectives of integration with industry standard methodologies and enabling largely waterfall management processes to work seamlessly with iterative development processes.

This chapter will also address the last remaining design objective extending the QFSD framework in order to support major business change programmes involving multiple projects, both locally and across multiple sites.

Figure 10.1 shows the components of the QFSD framework that are validated in this chapter.

![Figure 10.1: Framework validation and progress chart – Industry standard methodology](image)

Released Version 1.0 Page 241 of 422
10.1 Introduction

Having successfully completed the first two QFSD Framework introduction phases, this chapter now addresses phase three. Just as a reminder the remaining phases are:

- **Phase three**: Introduce industry standard methodologies and extend the QFSD framework coverage.

- **Phase four**: Extend the QFSD framework in order to support major business change programmes, both locally and across multiple sites.

Whilst the customised QFSD processes and associated repository provided a very comprehensive coverage of all the processes identified as required for the departments software development framework during phase two, it was further decided that The Stationary Office (2002) would be adopted as the standard project management methodology and that RUP (2007) would be adopted as the standard development methodology. The basis for this decision is described in section 10.3.1.

The reasons for adopting PRINCE2 were fourfold. Firstly, PRINCE2 has some similarities to the basic QFSD 'Organisational & Management Processes' and could be introduced following the now successful introduction of the basic QFSD framework. Secondly, PRINCE2 is in the public domain and hence recruiting additional project managers with similar experience would be easier especially across Europe as will be seen in phase four. Thirdly, given that Celesio is required to provide applications that interface to systems produced by the NHS, it would be part of the qualification process that those applications were delivered using a recognised industry standard; in this case a government recognised standard PRINCE2.

Fourthly, PRINCE2 is designed to be used with a variety of specialist methodologies and as such it should be possible to integrate it into the QFSD framework and with specialist methodologies such as RUP.

The decision to adopt the RUP methodology in the medium term came out of the department's early experience of using the QFSD 'clusterisation' technique. A number of projects were successfully delivered using the 'clusterisation' technique,
which gained wide acceptance within the new department and introduced the staff to iterative development. The similarity between RUP, which is one of the prime exponents of iterative development, and 'clusterisation' is such that the adoption of RUP would prove to be relatively straightforward.

Both PRINCE2 and RUP provide online repositories which, in the case of RUP specialist tools, made the integration with the main QFSD framework online application relatively straightforward. However, experience shows that change should be progressive and not overdone or it can have a detrimental effect (Orlikowski, 1993). Therefore the introduction of PRINCE2 and RUP were run as two separate pilot projects. It should be understood that the successful introduction of these two methods depended on the IT department having gained acceptance and experience using the QFSD framework, without this it would have been extremely difficult. In effect, phase three of the QFSD Framework introduction was divided into two steps. It should also be noted that the integration of PRINCE2 in to the QFSD Framework would provide all the necessary artefacts and processes required to extend the framework in order to meet the requirements of the CCTA Programme Management Standard (MSP) (CCTA, 1999). This will be further described as part of phase four later in the chapter.

10.2 Introduction of PRINCE2

If the reader is not familiar with the PRINCE2 methodology a short overview is provided by accessing the PRINCE2 web site, (www.ogc.gov.uk, 2002).

The PRINCE2 pilot involved two projects over a period of six months and was closely supported by the new project office. The participants in the project were given PRINCE2 training to practitioner level; this is, in fact, a formal and industry recognised project management qualification.

In terms of integrating PRINCE2 in to the QFSD Framework this consisted of three main activities:

- Adoption of the PRINCE2 artefact templates
- Review and integration of the PRINCE2 management processes
Assessment of the two projects to which the modified QFSD Framework was applied. The metrics assessed were the same as those used in the assessment of phase two in order to provide a good basis for comparison.

10.2.1 Adoption of the PRINCE2 templates

It should be noted that PRINCE2 provides a subset of templates that cover the QFSD Organisational and Management process areas only. PRINCE2 does not cover the specialist areas of Support and Execution.

From the template mapping shown in Table 10.1 it can be concluded that there is a reasonable overlap between the PRINCE2 and QFSD templates. A number of the PRINCE2 templates were adopted, whilst others were integrated within existing QFSD templates. This approach was taken in order to reduce disruption in the project teams and because the QFSD templates were much more detailed and prescriptive than the PRINCE2 templates, which, in a lot of cases, were very generic and lacked practical instruction on how they should be populated.

It should be noted that the following artefacts are designed as programme management artefacts by PRINCE2, but in fact already exist in the QFSD framework as part of the standard project management process.

- Project mandate
- Project brief
- Project organisation
- Exception reports
- Highlight reports
- Exception plans Project Board Approval
- Mid Stage Assessment
- Project-end notification
- Post project review
Table 10.1 PRINCE2 artefact template mapping

<table>
<thead>
<tr>
<th>QFSD Organizational Processes</th>
<th>PRINCE2 Replacement Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company Objectives</td>
<td>• No equivalent</td>
</tr>
<tr>
<td>Business Case</td>
<td>• Project Initiation Document (direct replacement)</td>
</tr>
<tr>
<td>Project Objectives</td>
<td>• Project Brief (direct replacement)</td>
</tr>
<tr>
<td>Project Approval Request.</td>
<td>• Project mandate (direct replacement)</td>
</tr>
<tr>
<td>Requirements Definition</td>
<td>• Specialist process (maintained QFSD Requirements Template)</td>
</tr>
<tr>
<td>Executive Test Plan Review</td>
<td>• No equivalent (maintained QFSD Executive Test Plan Template)</td>
</tr>
<tr>
<td>Executive Release Readiness</td>
<td>• Acceptance Criteria (merged Acceptance Criteria with QFSD Executive Release Readiness Template)</td>
</tr>
<tr>
<td>Process Improvements</td>
<td>• Project Quality Plan, Quality Log (merged Process Improvement Template with QFSD template)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Management Processes</th>
<th>Process Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Management Plan</td>
<td>• Project Initiation Document (direct replacement, but configured to use the agreed Celesio development lifecycle)</td>
</tr>
<tr>
<td>Project Planning</td>
<td>• Project plan (direct replacement)</td>
</tr>
<tr>
<td>Release Strategy</td>
<td>• Project Initiation Document (merged Project Initiation Document with QFSD template)</td>
</tr>
<tr>
<td>Resource Strategy</td>
<td>• Project Initiation Document (merged Project Initiation Document with QFSD template)</td>
</tr>
<tr>
<td>Requirements Capture</td>
<td>• Specialist process (maintained QFSD Requirements Template)</td>
</tr>
<tr>
<td>Development Process</td>
<td>• Specialist process (maintained QFSD Templates)</td>
</tr>
</tbody>
</table>
10.2.2 Adoption of PRINCE2 management processes

The PRINCE2 method is split into what it refers to as components and a set of eight processes (The Stationary Office, 2007). The components define the principles upon which the PRINCE2 method is built. The processes are a subset of the QFSD organisational and management processes. In terms of the eight PRINCE2 processes, Table 10.2 shows the mapping that was carried out between the QFSD lifecycle and PRINCE2. However, it is worth recalling that PRINCE2 only partially covers the organisational and management processes in QFSD at a very high level and does not address any of the specialist processes i.e. Support and Execution processes. These specialist processes will be partly covered when the RUP integration is considered later in the chapter. Therefore, the table maps the QFSD lifecycle (groupings of very detailed processes against the eight PRINCE2 processes).

Table 10.2 PRINCE2 process mapping

<table>
<thead>
<tr>
<th>QFSD lifecycle phases</th>
<th>PRINCE2 process model</th>
</tr>
</thead>
<tbody>
<tr>
<td>O – Start-up</td>
<td>SU – Project Start-up</td>
</tr>
<tr>
<td>A – Initiation and Planning</td>
<td>IP – Initiating a Project</td>
</tr>
<tr>
<td>----------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>PL – Planning</td>
<td></td>
</tr>
<tr>
<td>B – Requirements Capture</td>
<td>Managed artefacts and not project management controls</td>
</tr>
<tr>
<td>C – Functional Specification and high level design</td>
<td>Managed artefacts and not project management controls</td>
</tr>
<tr>
<td>D – Investment Appraisal</td>
<td>Unique to QFSD</td>
</tr>
<tr>
<td>E – Detailed Design, Code and Construction</td>
<td>Managed artefacts and not project management controls</td>
</tr>
<tr>
<td>F – Test</td>
<td>Managed artefacts and not project management controls</td>
</tr>
<tr>
<td>G - Implementation</td>
<td>MP – Manage Project Delivery</td>
</tr>
<tr>
<td>H – Close Down</td>
<td>CP – Close Project</td>
</tr>
<tr>
<td>X – All Phases</td>
<td>DP – Direct Project</td>
</tr>
<tr>
<td></td>
<td>CS – Controlling a Stage</td>
</tr>
<tr>
<td></td>
<td>SB – Manage Stage Boundaries</td>
</tr>
</tbody>
</table>

On the whole, the PRINCE2 eight processes are more than catered for in the QFSD Framework and are, in effect, a sub-set. Hence, by adopting the templates used by PRINCE2 and making some terminology changes, the PRINCE2 methodology was easily integrated into the QFSD Framework. This is not surprising given the level of sophistication of the QFSD organisational and management processes.

10.2.3 Pilot of PRINCE2 integration with QFSD

The modified QFSD Framework was used during two pilot projects with the metrics used being identical to those used during the analysis of phase two, thereby enabling a direct comparison in terms of improvement or otherwise.
Table 10.3 shows the first set of metrics based on schedule and budget predictability. The results show a decline in schedule predictability and budget accuracy of 3.9% and 2.3% respectively when compared with the results from phase two. This was attributed to the impact of changing the project management templates and modifying the existing processes.

Table 10.3 Phase three step one project delivery analysis

<table>
<thead>
<tr>
<th>Project Type</th>
<th>No of Lines of Code Changed</th>
<th>Schedule Success</th>
<th>Budget Success</th>
<th>Total Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoice matching (enhancement)</td>
<td>17,400 lines.</td>
<td>Planned duration 5 months. Actual 5.7 months.</td>
<td>10.1% Overspend.</td>
<td>24 Man Months.</td>
</tr>
<tr>
<td>Weldrick's new warehouse system (application from the market)</td>
<td>26,700 lines.</td>
<td>Planned duration 9 months. Actual 11.2 months.</td>
<td>12.4% Overspend.</td>
<td>43 Man Months.</td>
</tr>
</tbody>
</table>

Tables 10.4 and 10.5 take the first two development clusters for each project and records the chosen standard metric results based on the QFSD process performance method using accuracy of the clusters development estimates, quality of the required artefacts delivered by the cluster and errors discovered in project documentation due to the introduction of the Fagan (1986) inspection technique.

Table 10.4 Phase three step one cluster process performance analysis – Invoice Matching

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Process metric</th>
<th>% within 1 sigma</th>
<th>% within 2 sigma</th>
<th>% within 3 sigma</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Accuracy of estimates</td>
<td>87.4%</td>
<td>96.5%</td>
<td>98.2%</td>
</tr>
<tr>
<td>1</td>
<td>Quality of artefacts</td>
<td>85.4%</td>
<td>89.3%</td>
<td>93.2%</td>
</tr>
<tr>
<td>1</td>
<td>Errors discovered</td>
<td>83.0%</td>
<td>94.0%</td>
<td>97.3%</td>
</tr>
<tr>
<td>2</td>
<td>Accuracy of estimates</td>
<td>85.4%</td>
<td>94.1%</td>
<td>96.5%</td>
</tr>
<tr>
<td>2</td>
<td>Quality of artefacts</td>
<td>88.7%</td>
<td>91.3%</td>
<td>95.9%</td>
</tr>
</tbody>
</table>
The Tables 10.4 and 10.5 shows that the project execution and associated metrics are, for the most part, consistent across the project teams and that the adoption of the PRINCE2 project management method has not had a major positive or negative impact on the metrics recorded when compared with similar measurements carried out during phase two. This result is expected due to the close similarity of the QFSD management and PRICE2 management processes.

However, it can be seen that the initial clusters in each project showed a lower score than for phase two in the category: Quality of Artefacts. The reason being that in adopting a number of new templates it was found during inspections that although the contents of documents inspected maintained a high level of quality, mistakes in using the new templates were recorded. Again this was to be expected when any changes to established quality processes are made and shows that it is better to make small progressive stepwise changes, rather than taking a 'big bang' approach to process change (Gilb, 1981).

The QFSD process capability analysis considers all QFSD processes against which to measure the improving process capability of each project. The same two projects
as used for the process performance analysis have been considered and the results are shown in Tables 10.6 and 10.7.

Table 10.6 Phase two step one project process capability analysis – Invoice Matching

<table>
<thead>
<tr>
<th>Organisational Processes</th>
<th>Capability Level</th>
<th>Management Processes</th>
<th>Capability Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company Objectives</td>
<td>Level 1</td>
<td>Project Management Plan</td>
<td>Level 3</td>
</tr>
<tr>
<td>Business Case</td>
<td>Level 2</td>
<td>Project Planning</td>
<td>Level 2</td>
</tr>
<tr>
<td>Project Objectives</td>
<td>Level 2</td>
<td>Release Strategy</td>
<td>Level 2</td>
</tr>
<tr>
<td>Project Approval Request</td>
<td>Level 3</td>
<td>Resource Strategy</td>
<td>Level 2</td>
</tr>
<tr>
<td>Requirements Definition</td>
<td>Level 3</td>
<td>Requirements Capture</td>
<td>Level 1</td>
</tr>
<tr>
<td>Executive Test Plan Review</td>
<td>Level 2</td>
<td>Development Process</td>
<td>Level 2</td>
</tr>
<tr>
<td>Executive Release Readiness</td>
<td>Level 1</td>
<td>Risk Abatement</td>
<td>Level 1</td>
</tr>
<tr>
<td>Process Improvements</td>
<td>Level 3</td>
<td>Monitoring &amp; Control</td>
<td>Level 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Support Processes</th>
<th>Capability Level</th>
<th>Execution Processes</th>
<th>Capability Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Management</td>
<td>Level 2</td>
<td>Concept Definition</td>
<td>Level 1</td>
</tr>
<tr>
<td>Document Management</td>
<td>Level 2</td>
<td>Requirements Definition</td>
<td>Level 2</td>
</tr>
<tr>
<td>Quality Assurance</td>
<td>Level 1</td>
<td>Software Design</td>
<td>Level 2</td>
</tr>
<tr>
<td>Verification (inspection)</td>
<td>Level 2</td>
<td>Software Construction</td>
<td>Level 2</td>
</tr>
<tr>
<td>Validation</td>
<td>Level 2</td>
<td>Defect Repair Strategy</td>
<td>Level 1</td>
</tr>
<tr>
<td>Third Party Software &amp; Tools</td>
<td>Level 1</td>
<td>System Integration</td>
<td>Level 2</td>
</tr>
<tr>
<td>Defect Management</td>
<td>Level 2</td>
<td>Release Testing</td>
<td>Level 1</td>
</tr>
<tr>
<td>Process Improvement</td>
<td>Level 2</td>
<td>Organisational Readiness</td>
<td>Level 1</td>
</tr>
<tr>
<td>Change Control</td>
<td>Level 2</td>
<td>Software Maintenance</td>
<td>Level 1</td>
</tr>
</tbody>
</table>
The process capability analysis showed both an improvement in capability 'level' and a consistency in maintaining that level across both projects. It should be noted that the projects were executed in parallel and hence had different teams involved. It was expected that improvements would be found in the areas of 'Management Processes' as this is the area in which PRINCE2 provided new templates and modified project management processes and indeed this can be observed, but the improvement is not that significant. In fact, if we consider that the development teams were in to their third year of using the QFSD Framework, it could equally be considered that the improvements were in the maturity of applying the processes, rather than in the adoption of PRINCE2.

However, considering the reasons for the adoption of PRINCE2, which included the ability to recruit project managers skilled in its use and the requirement to use it from...
the NHS, then the integration and adoption of PRINCE2 went very smoothly and was a success.

10.2.4 Conclusions from the PRINCE2 pilot

In terms of meeting the overall objectives for adoption of PRINCE2 in to the QFSD Framework, the pilot was successful. However, it is worth highlighting the main points from the analysis, which confirm the original assumption that integration of PRINCE2 would be straightforward, but that care needed to be taken with any process change such that it did not have a negative impact.

- The PRINCE2 eight main processes turned out to be a small subset of the existing QFSD Framework Organisational and Management processes, of which there are nineteen. Where the PRINCE2 processes matched the QFSD processes they were integrated by modifying the QFSD process, rather than replacing them. The reason was that the QFSD processes not only matched, but significantly extended the PRINCE2 processes and provided much more detail on how to actually implement each process.

- In order to make sure that the PRINCE2 method was recognisable within the artefact deliverables the PRINCE2 document templates were adopted.

- Adoption of PRINCE2 caused an initial impact on projects showing a decline in schedule and budgetary predictability of 3.9% and 2.3% respectively. This was traced to the adoption of the new templates and was addressed in subsequent projects as the teams became more familiar with the new templates and changes in project monitoring.

- Measuring process performance for a selected number of process over two projects showed a negligible impact, which was as expected given the closeness of the QFSD and PRINCE2 management processes in the areas that they overlapped.

- Measuring process capability again showed a negligible impact, except for the obvious drop in capability measure in the quality artefacts. This drop was not in
the quality of the content, but in terms of the construction and management of the new templates. Both projects included this in the post-project review and it was subsequently addressed.

Monitoring of subsequent projects at Celesio showed a sustained improvement in scheduled software deliveries, costs and overall quality. However, these improvements were not really significant when compared to the previous metrics for Celesio projects that used the pure QFSD framework. However, as stated previously, adoption of PRINCE2 enabled project managers from the market to be recruited quickly in order to reach the goal of having industry accredited project managers. This overall improvement in recognised project management skills was not only seen in terms of the NHS requirements, but also in the opening pharmaceutical market concerning manufacturers' delivery directly to pharmacies, which could reduce Celesio's product volumes. Wholesalers wishing to act on behalf of manufacturers would need to show a high level of quality in order to become part of their extended supply chain.

10.3 Introduction of RUP

If the reader is not familiar with the RUP methodology a short overview is provided by the RUP website (http://www.rational.com/products/rup, 2007).

In this second step of phase three, the RUP integration pilot involved two projects over a period of fourteen months which were closely supported by the new project office. The participants in the project were given commercially available RUP basic training.

The integration RUP into the QFSD Framework was more challenging as it needed to integrate within the QFSD Support and Execution process, but also it needed to interface with PRINCE2.

The integration activities consisted of two main activities:

- Review and integrate the RUP process model into QFSD & PRINCE2
• Assess the two projects to which the extended QFSD Framework was applied. The metrics assessed are the same as those used in the assessment of phase two, and in the PRINCE2 pilots, in order to provide a good basis for comparison.

10.3.1 Review and integration of RUP

QFSD is designed to cover all aspects of software management and development in a thorough and prescriptive fashion. This has been proven as a sound approach given the extensive application of the QFSD framework both in Fisher-Rosemount and Celesio. The approach is based on the very simple principle that to obtain consistency of process the processes need to be fully documented and unambiguous. When faced with the integration of an industry standard methodology, which replaces a subset of the QFSD processes, then a strategy for this must be adopted. The strategy adopted is as follows:

• **Keep it simple.** The integrated process should be understandable to both the QFSD / PRINCE2 and RUP communities.

• **Make it "do-able."** The process should be easy to implement in a short time frame with the minimum amount of effort.

• **Maintain separation of concerns.** RUP focuses on the development of software products, thus it should be a complete replacement of the QFSD execution processes in order not to muddy these distinctions.

• **Maintain flexibility.** The integrated process should permit additional methods to be used with QFSD and RUP.

• **Create synergy.** The integration should result in a "best of both worlds" solution that emphasizes each method's strengths.

Using these principles means that a loosely coupled integration strategy is the best approach for integration of RUP. QFSD covers organisational, management, execution and support processes. PRINCE2, as explained earlier in the Chapter, covers mainly management processes, with some overlap in terms of organisational processes. RUP, on the other hand covers mainly execution processes, with some overlap of management processes. It also provides tools, techniques and roles for
performing software execution, plus some project management roles, artefacts and guiding principles. Therefore, it overlaps to a limited extent with QFSD and PRINCE2 in terms of management processes and with QFSD in terms of the execution processes. However, the project management processes were reviewed and found to be limited when compared to the now enhanced QFSD management processes and hence will not be considered further in the integration, other than to apply some direct development management at the software development team level. Figure 10.2 shows a pictorial view of the proposed QFSD, PRINCE2 and RUP process mapping.

Figure 10.2: QFSD, PRINCE2 & RUP Mapping

As can be seen from the Figure 10.2 RUP is being considered mainly in the context of the execution processes and hence will need to be integrated with the remaining QFSD processes. The categories selected in order to evaluate the integration of RUP are as follows:

- Terminology
- Guiding principles
- Roles
- Deliverables, products and artefacts
- Tools and techniques

<table>
<thead>
<tr>
<th>Organisational Processes</th>
<th>PRINCE2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management Processes</td>
<td></td>
</tr>
<tr>
<td>QFSD</td>
<td></td>
</tr>
<tr>
<td>Execution Processes</td>
<td></td>
</tr>
<tr>
<td>RUP</td>
<td></td>
</tr>
<tr>
<td>Support Processes</td>
<td></td>
</tr>
</tbody>
</table>

Figure 10.2: QFSD, PRINCE2 & RUP Mapping
10.3.1.1 Terminology

Terminology is part of the identity of any methodology defining the language used by the community that use it. After an extensive evaluation of all three methodologies, there are no terms that are used in the methodologies that are associated with different and contradictory meanings. From this point of view the methodologies are compatible.

There are, however different terms used for the same or very similar concepts. For example, PRINCE2 describes physical items as Products, whereas RUP and QFSD call them Artefacts. Both QFSD and PRINCE2 use the term Processes, whereas RUP uses the term Disciplines. Whilst the QFSD and PRINCE2 terminology differences could be easily balanced, the differences between the RUP terminology and both QFSD and PRINCE2 is such that it was decided to keep the terminologies separate rather than attempting to create a combined terminology. This approach is also recommended for the following reasons:

- There are a limited number of roles that need to use both sets of terminology and hence the impact on maintaining both sets of terminology is limited.
- Keeping the RUP terminology will make it easier to integrate new members of staff that have had previous experience with the methodology.
- RUP is likely to continue to evolve. For example, RUP now uses the term "Discipline" for the concept that it used to call "Core Workflow".

10.3.1.2 Guiding principles

Both QFSD and PRINCE2 have a set of programme related processes that do not appear in RUP. These include:

- The project management organisation, specifically the project board that manages exceptions and represents the interests of the sponsor, users and the supplier.
- The project managers delegated responsibility, beyond which a project exception is raised.
The management of the customer and various suppliers.

In terms of management and support processes, RUP does not include the following and hence they are provided by the QFSD framework:

- Configuration management planning.
- Formal document management method.
- Active quality assurance guidance. RUP requires a quality plan to be in place, but does not define the details on how to do this.
- Verification in terms of inspection. Again this is required by RUP, but not detailed on how to carry it out.
- Management of third party software and reuse.
- Defect management.
- Process improvement.
- Project review process.
- Detailed project monitoring and control.

RUP does have the concept of a business case, but the QFSD business case is already established. Changing the business case given that the business users are familiar with the existing template and approach would have had a negative impact.

At the management process level the following guiding principles are prescribed by all three methodologies in ways that are compatible and complementary:

- Product based (or component based) planning
- Iterative planning
- Managing change
- Managing risk
- Managing quality
The following are defined in slightly different ways, and these differences need modification to the overall lifecycle model:

- **Phased approach vs. iterative process**

QFSD proposes that a project should be broken down into a number of management and controllable phases (stages is the term used on PRINCE2, but they have already been mapped into the QFSD phases during its integration). The RUP process model prescribes four main phases (Inception, Elaboration, Construction and Transition), with at least one iteration of each phase, depending on the nature of the project. The recommended way to map QFSD and RUP is to equate the RUP phases to the QFSD iterative phases ((A) through (F)) as defined in Figure 10.3. It should be remembered that the 'clusterisation' approach applied in the Core QFSD Framework promotes the idea of running multiple clusters in parallel, each of which can be in a different phase, or can repeat any phase multiple times. Hence the RUP iterative phases can be mapped into the QFSD phases, but still use the RUP methodology, i.e. QFSD treats RUP as a replacement for its Execution processes, but maintains the same interface between the Organisational, Management and Supporting processes as it had with its own Execution processes.

![Figure 10.3: Alignment of RUP Phases with QFSD Phases](image-url)
Table 10.8 defines in more detail the QFSD Lifecycle phase to RUP phase mapping that were implemented during the integration.

### Table 10.8 QFSD Phase to RUP Phase Mapping

<table>
<thead>
<tr>
<th>QFSD Lifecycle Phase</th>
<th>RUP Phase Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - Start Up</td>
<td>QFSD and PRINCE2 prescribe start-up for defining the project's scope, objectives, management organization and a distinct project initiation phase for creating the Project Initiation Document (PID). This phase also includes the definition of the business case and hence the projects return on investment objectives (ROI) and contribution to the overall company's business and IT strategies. RUP does have the concept of business and project vision definitions, but these are far outweighed by the existing processes in QFSD.</td>
</tr>
<tr>
<td>A - Initiation &amp; Planning</td>
<td>RUP does not have a distinct project initiation, but incorporates various aspects of project initiation at the beginning of each inception phase, which is the first part of each subsequent iteration. However, within the inception phase RUP has project definition, planning, environment preparation, which includes guidelines and tool definitions. At this level the RUP processes have been integrated in to the QFSD Execution process level in terms of management of the execution of each development iteration. Therefore the initiation and planning phase in QFSD has not been modified significantly as it already has a management interface between its management processes and execution processes, which are already iterative in nature.</td>
</tr>
<tr>
<td>QFSD Lifecycle Phase</td>
<td>RUP Phase Mapping</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>B - Requirements Capture</td>
<td>The requirements phase and templates in the QFSD Framework are a direct replacement for the RUP templates and hence it was less impact to continue to use them.</td>
</tr>
<tr>
<td>C - Functional Specification &amp; High Level Design</td>
<td>The templates provided by RUP were adopted as an option set in the QFSD repository.</td>
</tr>
<tr>
<td>D - Investment Appraisal</td>
<td>This is a QFSD management stage and although it is possible to apply it during each iteration, in reality it is usually only applied once in any given project, unless there is a large change in project scope.</td>
</tr>
<tr>
<td>E - Detailed Design, Code &amp; construction</td>
<td>The processes and templates provided by RUP were adopted as an option set in the QFSD repository.</td>
</tr>
<tr>
<td>F - Test</td>
<td>The processes and templates provided by RUP were adopted as an option set in the QFSD repository.</td>
</tr>
<tr>
<td>G - Implementation</td>
<td>The processes and templates provided by RUP were adopted as an option set in the QFSD repository.</td>
</tr>
<tr>
<td>H - Close Down</td>
<td>The QFSD process was retained as both RUP and QFSD processes were very similar.</td>
</tr>
<tr>
<td>X - All phases</td>
<td>The processes in this phase cover all basic project management and quality management processes and as such retained the QFSD Framework Core processes.</td>
</tr>
</tbody>
</table>

Table 10.8 indicates the overall level to which RUP was integrated into QFSD Framework. As described earlier, the objective was to adopt the RUP development
processes and templates where possible. However, again the QFSD Framework provides the management structure and quality assurance processes around the adopted methodology to ensue that it can be quickly and flexibly adopted, causing minimum disruption to the overall level of quality and delivery timescales and to preserve the community around the adopted methodology.

10.3.1.3 Roles and Responsibilities

The role definitions in QFSD and RUP are largely compatible, since they have very little overlap.

At the Organisational and Management process levels in QFSD Framework the high level roles are defined in terms of the Project Board:

- The Sponsor representing the interests of the investor
- The Senior User, representing the interests of the users that deliver the business benefits
- The Senior Supplier, representing the interests of the software supplier

The above roles do not exist in RUP.

At the project management level the main role defined by QFSD, PRINCE2 and RUP is that of the Project Manager. QFSD also prescribes the roles of Project Support and Project Assurance, which can be equated to the role of Project Reviewer in RUP.

There is also an overlap between some of the responsibilities of the Project Manager in QFSD and some of the specialist roles in RUP:

- Defining and designing the project workflows
- Selecting relevant specialist products or artefacts
- Defining guidelines, standards and templates

These are assigned as the responsibility of the Process Engineer in RUP. In practice these were divided between the Process Engineer and the Project Manager, with the
Project Manager maintaining the 'Defining Guidelines, standards and templates', which is a strong part of the QFSD Framework.

Two further responsibilities were found in RUP, which were kept under the control of the Project Manager:

- Developing the business vision (RUP: Business Process Analyst)
- Developing the Project Vision (Systems Analyst)

Table 10.9 shows the final combined set of roles used.

Table 10.9 Combined RUP and QFSD Roles

<table>
<thead>
<tr>
<th>Process Level</th>
<th>QFSD</th>
<th>RUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organisational</td>
<td><strong>Project Board</strong></td>
<td>Stakeholders</td>
</tr>
<tr>
<td></td>
<td>• Executive</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Programme Director</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Programme Manager</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Business Change Manager</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Senior User</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Senior Supplier</td>
<td></td>
</tr>
<tr>
<td>Management</td>
<td>Project Manager</td>
<td>Project Manager</td>
</tr>
<tr>
<td></td>
<td>Project Assurance</td>
<td>Project Reviewer</td>
</tr>
<tr>
<td></td>
<td>Project Support</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Business Analyst</td>
<td></td>
</tr>
<tr>
<td>Support</td>
<td>Project Office Support</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Quality Assurance Manager</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reuse Coordinator</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Change Control Manager</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Configuration Manager</td>
<td></td>
</tr>
<tr>
<td>Execution</td>
<td><strong>Management Roles</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Team Leader</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Release Manager</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Build Manager</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Test Manager</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Management Roles</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Process Engineer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Configuration Manager</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Change Control Manager</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Deployment Manager</td>
<td></td>
</tr>
<tr>
<td>Process Level</td>
<td>QFSD</td>
<td>RUP</td>
</tr>
<tr>
<td>---------------</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Analyst Roles</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Business Designer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Business Model Reviewer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Business Process Analyst</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Reviewer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Specifier</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Systems Analyst</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• User Interface Designer</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Development Roles</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Architecture Reviewer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Capsule Designer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Code Reviewer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Database Designer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Design Reviewer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Designer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Implemener</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Integrator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Software Architect</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Test Roles</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Test Designer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Tester</td>
</tr>
</tbody>
</table>

- **Development Roles**
  - System Architect
  - Infrastructure Specialist
  - Database / Repositories Administrator
  - Design Reviewer
  - Code Developer
  - Code Reviewer
  - Project Librarian
  - Defect Coordinator
  - System Administrator
  - Technical Writer

- **Test Roles**
  - Test Designer
  - Tester
### 10.3.1.4 Deliverables, products and artefacts

RUP does produce a limited number of project management artefacts. However, the majority of these have been replaced by the combined QFSD and PRINCE2 artefacts. However, “Iteration Assessment” and “Iteration Plan” have been retained for the management of iterations.

In addition, the RUP “Software Development Plan” and “Risk Management Plan” are replaced by the combined QFSD and PRINCE2 artefact “Project Management Plan”.

The combined set of products and artefacts is shown in Table 10.10.

#### Table 10.10 Combined RUP and QFSD / PRINCE2 products and artefacts

<table>
<thead>
<tr>
<th>QFSD / PRINCE2 Organisational Processes</th>
<th>RUP Artefacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company Objectives</td>
<td>• No equivalent</td>
</tr>
<tr>
<td>Business Case</td>
<td>• Business Case, Business Vision &amp; Project Vision (QFSD / PRINCE2 business case Template maintained)</td>
</tr>
<tr>
<td>Project Objectives</td>
<td>• Project vision (QFSD / PRINCE2 Project Management Plan template maintained)</td>
</tr>
</tbody>
</table>

*Note: Shaded areas indicate roles included in the combined roles model*
<table>
<thead>
<tr>
<th>Project Approval Request</th>
<th>• No equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Definition</td>
<td>• Specialist process (maintained QFSD / PRINCE2 Requirements Template)</td>
</tr>
<tr>
<td>Executive Test Plan Review</td>
<td>• No equivalent (maintained QFSD / PRINCE2 Executive Test Plan Template)</td>
</tr>
<tr>
<td>Executive Release Readiness</td>
<td>• Product acceptance plan (maintained QFSD / PRINCE2 template)</td>
</tr>
<tr>
<td>Process Improvements</td>
<td>• Quality assurance plan (maintained QFSD / PRINCE2 template)</td>
</tr>
</tbody>
</table>

### QFSD / PRINCE2 Management Processes

<table>
<thead>
<tr>
<th>QFSD / PRINCE2 Management Processes</th>
<th>RUP Artefacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Management Plan</td>
<td>• Software development plan, Problem resolution plan, Review Record &amp; Status Assessment (maintained QFSD / PRINCE2 template)</td>
</tr>
<tr>
<td></td>
<td>• Iteration Plan and Iteration Assessment (integrated in to the Project Management Plan)</td>
</tr>
<tr>
<td>Project Planning</td>
<td>• Iteration Plan (Part of the Project Management Plan template)</td>
</tr>
<tr>
<td>Release Strategy</td>
<td>• Product Acceptance Plan (part of the Project Management Plan QFSD / PRINCE2 template)</td>
</tr>
<tr>
<td>Resource Strategy</td>
<td>• Iteration Plan (part of the Project Management Plan QFSD / PRINCE2 template)</td>
</tr>
<tr>
<td>Requirements Capture</td>
<td>• Business Modelling (maintained QFSD / PRINCE2 Requirements Template, but adopted the business modelling templates)</td>
</tr>
<tr>
<td>Development Process</td>
<td>• Specialist process (updated QFSD / PRINCE templates to factor in RUP iterative approach)</td>
</tr>
<tr>
<td>Risk Management</td>
<td>• Risk Management Plan and Risk Log (maintained QFSD / PRINCE2 risk templates)</td>
</tr>
<tr>
<td>Monitoring &amp; Control</td>
<td>• Iteration assessment (maintained QFSD / PRINCE2 monitoring templates)</td>
</tr>
<tr>
<td>Project Review</td>
<td>• Iteration Assessment (maintained QFSD / PRINCE2 stage review templates)</td>
</tr>
<tr>
<td>Test Management</td>
<td>• Specialist process (adopted RUP Test Artefact Set)</td>
</tr>
<tr>
<td>Problem Resolution</td>
<td>• Problem resolution plan (maintained QFSD Template)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>QFSD / PRINCE2 Support Processes</th>
<th>RUP Artefacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Management</td>
<td>• Configuration &amp; Change Management (maintained QFSD / PRINCE2 configuration and change control templates)</td>
</tr>
<tr>
<td>Document Management</td>
<td>• No equivalent</td>
</tr>
<tr>
<td>Quality Assurance</td>
<td>• Quality Assurance Plan (maintained QFSD / PRINCE2 quality plan templates)</td>
</tr>
<tr>
<td>Verification (inspection)</td>
<td>• Quality Assurance Plan (maintained QFSD / PRINCE2 quality plan templates)</td>
</tr>
<tr>
<td>Validation</td>
<td>• Quality Assurance Plan (maintained QFSD / PRINCE2 quality plan templates)</td>
</tr>
<tr>
<td>Third Party Software &amp; Reuse</td>
<td>• No equivalent</td>
</tr>
<tr>
<td>Defect Management</td>
<td>• Implementation Artefact Set (adopted RUP templates, except for the statistical monitoring provided by QFSD)</td>
</tr>
<tr>
<td>Process Improvement</td>
<td>• No equivalent</td>
</tr>
<tr>
<td>Change Control</td>
<td>• Configuration &amp; Change Management (maintained QFSD / PRINCE2 configuration and change control templates)</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>QFSD / PRINCE2 Implementation Processes</th>
<th>RUP Artefacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approach Definition</td>
<td>• Software development plan, Problem resolution plan, Review Record &amp; Status Assessment (maintained QFSD / PRINCE2 template)</td>
</tr>
<tr>
<td>Requirements Definition</td>
<td>• Specialist process (maintained QFSD / PRINCE2 Requirements Template)</td>
</tr>
<tr>
<td>Software Design</td>
<td>• Analysis and Design Artefact Set</td>
</tr>
<tr>
<td>Software Construction</td>
<td>• Implementation Artefact Set</td>
</tr>
<tr>
<td>Defect Repair Strategy</td>
<td>• No equivalent</td>
</tr>
<tr>
<td>System Integration</td>
<td>• No equivalent</td>
</tr>
<tr>
<td>Release Testing</td>
<td>• Test Artefact Set</td>
</tr>
<tr>
<td>Organizational Readiness</td>
<td>• Deployment Artefact Set</td>
</tr>
<tr>
<td>Software Maintenance</td>
<td>• No equivalent</td>
</tr>
</tbody>
</table>

10.3.1.5 Tools & Techniques

The adoption of the RUP methodology in terms of tools was relatively straightforward. RUP is supported by an online web-based repository in which all templates are held. The approach was to firstly modify the QFSD online repository to include the RUP templates and change the relevant navigation to use the RUP templates and secondly, reflect changes in the overall lifecycle in order to show how RUP was integrated into the overall QFSD framework as described in the earlier sections.
10.3.2 Pilot of RUP integration with QFSD

The modified QFSD Framework was used during two further pilot projects with the metrics used being identical to those used during the PRINCE integration, again enabling a direct comparison in terms of improvement or otherwise.

Table 10.11 shows the first set of metrics based on schedule and budget predictability. The results show an improvement in schedule achievement and budget accuracy of 0.65% and 0.8% respectively when compared with the results from the PRINCE2 adoption. From the comparison it would seem that introduction of the RUP methodology did not have a significant impact. However, it was found from both the process performance and process capability analysis that the introduction of RUP did have some detrimental effects on a number of processes. However, the minimal impact on the overall schedule and budget predictability was due to a general overall improvement in the project team's skills concerning the use of the QFSD framework since the time when the PRINCE2 pilot was carried out. A steady improvement over a further seven projects, following the PRINCE2 pilot, showed that schedule and budget predictability was improving after every project.

Table 10.11 Phase three step two project delivery analyses

<table>
<thead>
<tr>
<th>Project Type</th>
<th>No of Lines of Code Changed</th>
<th>Schedule Success</th>
<th>Budget Success</th>
<th>Total Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer Profitability (new in-house development)</td>
<td>21,300 lines.</td>
<td>Planned duration 12 months.</td>
<td>12.2% Under spend.</td>
<td>42.6 Man Months.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Actual 13.6 months.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Win FSM (new in-house development)</td>
<td>19,700 lines.</td>
<td>Planned duration 11 months.</td>
<td>9.1% Under spend.</td>
<td>28.5 Man Months.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Actual 12.2 months.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tables 10.12 and 10.13 take the first two development clusters for each project as before, and records the chosen standard metric results based on:

- the QFSD process performance method using accuracy of the clusters development estimates
- the quality of the required artefacts delivered by the cluster
- the errors discovered in project documentation due to the introduction of the Fagan (1986) inspection technique.

Table 10.12 Phase three step two cluster process performance analysis – Customer Profitability

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Process metric</th>
<th>% within 1 sigma</th>
<th>% within 2 sigma</th>
<th>% within 3 sigma</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Accuracy of estimates</td>
<td>80.4%</td>
<td>89.5%</td>
<td>93.2%</td>
</tr>
<tr>
<td>1</td>
<td>Quality of artefacts</td>
<td>84.1%</td>
<td>88.3%</td>
<td>91.4%</td>
</tr>
<tr>
<td>1</td>
<td>Errors discovered</td>
<td>81.0%</td>
<td>89.2%</td>
<td>93.3%</td>
</tr>
<tr>
<td>2</td>
<td>Accuracy of estimates</td>
<td>83.2%</td>
<td>91.2%</td>
<td>95.1%</td>
</tr>
<tr>
<td>2</td>
<td>Quality of artefacts</td>
<td>85.5%</td>
<td>90.0%</td>
<td>92.4%</td>
</tr>
<tr>
<td>2</td>
<td>Errors discovered</td>
<td>82.6%</td>
<td>90.4%</td>
<td>93.9%</td>
</tr>
<tr>
<td>3</td>
<td>Accuracy of estimates</td>
<td>93.0%</td>
<td>97.8%</td>
<td>99.9%</td>
</tr>
<tr>
<td>3</td>
<td>Quality of artefacts</td>
<td>91.5%</td>
<td>97.7%</td>
<td>98.6%</td>
</tr>
<tr>
<td>3</td>
<td>Errors discovered</td>
<td>90.6%</td>
<td>96.5%</td>
<td>98.1%</td>
</tr>
</tbody>
</table>

Table 10.13 Phase three step two cluster process performance analysis – Win FSM

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Process metric</th>
<th>% within 1 sigma</th>
<th>% within 2 sigma</th>
<th>% within 3 sigma</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Accuracy of estimates</td>
<td>82.5%</td>
<td>89.3%</td>
<td>94.2%</td>
</tr>
<tr>
<td>1</td>
<td>Quality of artefacts</td>
<td>84.1%</td>
<td>89.4%</td>
<td>91.6%</td>
</tr>
<tr>
<td>1</td>
<td>Errors discovered</td>
<td>81.7%</td>
<td>89.0%</td>
<td>92.2%</td>
</tr>
<tr>
<td>2</td>
<td>Accuracy of estimates</td>
<td>83.5%</td>
<td>92.6%</td>
<td>96.4%</td>
</tr>
<tr>
<td>2</td>
<td>Quality of artefacts</td>
<td>83.0%</td>
<td>88.4%</td>
<td>89.8%</td>
</tr>
<tr>
<td>2</td>
<td>Errors discovered</td>
<td>84.7%</td>
<td>92.7%</td>
<td>94.5%</td>
</tr>
<tr>
<td>3</td>
<td>Accuracy of estimates</td>
<td>94.6%</td>
<td>98.2%</td>
<td>98.9%</td>
</tr>
<tr>
<td>3</td>
<td>Quality of artefacts</td>
<td>92.1%</td>
<td>97.9%</td>
<td>98.7%</td>
</tr>
<tr>
<td>3</td>
<td>Errors discovered</td>
<td>93.0%</td>
<td>97.3%</td>
<td>98.7%</td>
</tr>
</tbody>
</table>
The projects shown in the tables were executed by different project teams in order to verify the impact of the RUP methodology's introduction and to eliminate cross project learning. As can be seen from the tables the overall process performance for the metrics recorded was again very high and indicates that the introduction of RUP in future projects would not have a large detrimental impact. However, as with the introduction of the PRINCE2 methodology, it can be seen that the metrics for the first cluster shows an initial decline in performance, which then quickly recovers in the subsequent clusters. In this case the clusters were executed sequentially and used the same resources in order to monitor the early usage impact and time for adjustment to the new methodology.

Again, the observation is that no matter what early training is given concerning a process change, there will always be an impact when the new process is first put into practice. However, in this case the impact was small and one could further suggest that introduction of similar processes to those already used within a project, where the project team are experienced in the application of the previous process, results in only a negligible impact.

The QFSD process capability analysis considers all QFSD processes against which to measure the improving process capability of each project. The same two projects as used for the process performance analysis have been considered and the results are shown in Tables 10.14 and 10.15.

Table 10.14 Phase two step two project process capability analysis – Customer Profitability

<table>
<thead>
<tr>
<th>Organisational Processes</th>
<th>Capability Level</th>
<th>Management Processes</th>
<th>Capability Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company Objectives</td>
<td>Level 2</td>
<td>Project Management Plan</td>
<td>Level 3</td>
</tr>
<tr>
<td>Business Case</td>
<td>Level 2</td>
<td>Project Planning</td>
<td>Level 3</td>
</tr>
<tr>
<td>Project Objectives</td>
<td>Level 2</td>
<td>Release Strategy</td>
<td>Level 2</td>
</tr>
<tr>
<td>Project Approval Request</td>
<td>Level 3</td>
<td>Resource Strategy</td>
<td>Level 2</td>
</tr>
<tr>
<td>Requirements Definition</td>
<td>Level 2</td>
<td>Requirements Capture</td>
<td>Level 2</td>
</tr>
<tr>
<td>Executive Test Plan Review</td>
<td>Level 2</td>
<td>Development Process</td>
<td>Level 2</td>
</tr>
<tr>
<td>Executive Release Readiness</td>
<td>Level 2</td>
<td>Risk Abatement</td>
<td>Level 3</td>
</tr>
<tr>
<td>Process Improvements</td>
<td>Level 3</td>
<td>Monitoring &amp; Control</td>
<td>Level 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Project Review</td>
<td>Level 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Test Management</td>
<td>Level 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Problem Resolution</td>
<td>Level 2</td>
</tr>
</tbody>
</table>
Table 10.15 Phase two step one project process capability analysis – Win FSM

<table>
<thead>
<tr>
<th>Support Processes</th>
<th>Capability Level</th>
<th>Execution Processes</th>
<th>Capability Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Management</td>
<td>Level 2</td>
<td>Concept Definition</td>
<td>Level 2</td>
</tr>
<tr>
<td>Document Management</td>
<td>Level 2</td>
<td>Requirements Definition</td>
<td>Level 2</td>
</tr>
<tr>
<td>Quality Assurance</td>
<td>Level 2</td>
<td>Software Design</td>
<td>Level 2</td>
</tr>
<tr>
<td>Verification (inspection)</td>
<td>Level 2</td>
<td>Software Construction</td>
<td>Level 2</td>
</tr>
<tr>
<td>Validation</td>
<td>Level 2</td>
<td>Defect Repair Strategy</td>
<td>Level 2</td>
</tr>
<tr>
<td>Third Party Software &amp; Raw</td>
<td>Level 1</td>
<td>System Integration</td>
<td>Level 2</td>
</tr>
<tr>
<td>Defect Management</td>
<td>Level 3</td>
<td>Release Testing</td>
<td>Level 2</td>
</tr>
<tr>
<td>Process Improvement</td>
<td>Level 2</td>
<td>Organisational Readiness</td>
<td>Level 2</td>
</tr>
<tr>
<td>Change Control</td>
<td>Level 3</td>
<td>Software Maintenance</td>
<td>Level 2</td>
</tr>
</tbody>
</table>

Given that the development teams involved in both projects have, since the integration of the PRINCE2 methodology, been using QFSD on a significant number of other projects, it is not surprising that the capability level in a number of areas has improved in general. However, the capability in the areas of development process, software design and software construction have not improved and remain at level 2.

In previous projects using the Core QFSD processes, including the development process known as ‘clusterisation’, level 3 had been achieved on a number of projects.
However, with the adoption of RUP the early adopter projects have dropped back to level 2. However, following the pilot, further projects were executed using the RUP extension and again level 3 was achieved. The observation was that with the adoption of a new methodology, experience needed to be gained by both the development teams and the project management teams in terms of understanding the new process and the best points at which to monitor the new and extended development processes.

10.3.3 Conclusions from the RUP pilot

In terms of meeting the overall objectives for adoption of PRINCE2 in to the QFSD Framework, the pilot was again a success. However, having completed the PRINCE2 methodology integration, an adoption strategy was proposed for the integration of the RUP methodology. Whilst the RUP methodology has been integrated and is working in practice, it is worth considering the extent to which the actual integration activities matched the proposed adoption strategy.

To recap the adoption strategy was as follows:

- **Keep it simple.** The integrated process should be understandable to both the QFSD / PRINCE2 and RUP communities.
- **Make it "do-able."** The process should be easy to implement in a short time frame with the minimum amount of effort.
- **Maintain separation of concerns.** RUP focuses on the development of software products, thus it should be a complete replacement of the QFSD execution processes in order not to muddy these distinctions.
- **Maintain flexibility.** The integrated process should permit additional methods to be used with QFSD and RUP.
- **Create synergy.** The integration should result in a "best of both worlds" solution that emphasizes each method's strengths.

Keep it simple: In general this was achieved as the RUP pilot showed that during the initial cluster the project team quickly learned how to apply the RUP extensions
and then began to improve the efficiency with which they were applied during subsequent clusters.

Make it “do-able”: There are two perspectives on this part of the strategy. Application of the new RUP processes was easily achieved by the project team. However, there was a learning curve for the project managers in terms of understanding how the new RUP processes integrated with the existing QFSD processes. This was not a significant problem, but rather reinforced the need for training in any changes in methodology, as quickly as possible for all parties involved with a project.

Maintain separation of concerns: This was not cleanly executed in the areas of development roles and development artefacts. In terms of development roles a mixture of QFSD and RUP roles were adopted. However, this approach did not cause a problem as development roles are well understood by developers and were easy to understand and adopt. In terms of the execution processes both the test and implementation artefacts were adopted directly from RUP. However, what did cause some difficulty initially was the failure to adopt the RUP software development plan process, which ended up having to be partially added to the existing QFSD development process.

Maintain flexibility: Given that separation of concerns was not kept clean, then adoption of another development methodology would require detailed integration as can be seen from the design effort required to integrate RUP.

Create synergy: Whilst the integration approach did not entirely adhere to maintaining separation of concerns, it did optimise QFSD, PRINCE2 and RUP. So in that respect it did provide the “best of both worlds”.

10.4 Conclusions

This chapter addressed the third phase of introducing the QFSD Framework into Celesio AG, following the successful application of the methodology in the highly commercial and fast moving business environment within Celesio AG. This validated both its applicability and scalability, as described in the previous chapter.
The third phase of validating the QFSD Framework involved testing whether it could be integrated with other industry standard methodologies. Therefore the chapter addresses a further two QFSD design principles which are:

- Integration with industry standard methodologies
- Enabling largely waterfall management processes to work seamlessly with iterative development processes

The second objective was included in the QFSD design principles as generally there is a split between management methodologies and development methodologies, where the former are usually waterfall model based and the latter follow an iterative model.

As with the nature of this thesis the approach to integration of industry standard methodologies was proven by applying the integrated processes to actual software projects and assessing the overall impact using a set of standard metrics, generated from the QFSD frameworks own tool-set. The two methodologies chosen had to cover both a waterfall and iterative methodology. In addition, they had to be methodologies that Celesio AG wanted to adopt in order that real projects could be used to validate the integration.

The two methodologies selected were PRINCE2 and the Rational Unified Process (RUP) for reasons explained earlier in the chapter.

The general approach used for the integration of PRINCE2 was as follows:

- Compare and integrate the QFSD processes with the eight top level PRINCE2 management processes
- Review and adopt where appropriate the PRINCE2 artefact templates
- Use the extended QFSD framework as the basis for two real projects.
- Apply a standard set of key performance and capability metrics in order to measure the improvement or otherwise resulting from the use of the extended methodology.

The general approach used for the integration of RUP was as follows:
Compare and integrate the RUP process model into QFSD & PRINCE2. Whilst with the PRINCE2 methodology this was simply a comparison as the PRINCE2 processes were really a sub-set of the QFSD management processes, the integration of RUP required a more rigorous approach. Therefore the following criteria were considered during the integration process:

- Terminology: ensure that there are no misunderstandings between artefact naming between the methodologies
- Guiding principles: ensure that the integration of the new methodology is easy to understand and to use
- Roles: make sure that roles are clearly understood and to eliminate any duplicate or overlapping roles between the methodologies
- Deliverables, products and artefacts: review and adopt where appropriate the RUP artefacts
- Tools and techniques: as RUP is provided with its own online repository, assess if this should be integrated, or if the QFSD remains as the master repository

Apply a standard set of key performance and capability metrics in order to measure the improvement or otherwise resulting from the use of the extended methodology.

The standard set of metrics applied to each of the four pilot projects covered general project performance and specific project capability metrics as listed below:

- Overall schedule performance against planned schedule performance
- Overall budget performance against planned budget performance
- Accuracy of 'cluster' estimates against actual values
- Quality of project artefacts delivered by each 'cluster'
- Number and level of errors discovered during the lifecycle of each 'cluster'
As can be seen from the results of applying the PRINCE2 extended QFSD framework to two major software projects and, subsequently, the combined PRINCE2 and RUP extended framework to a further two software projects, the integration was successful with little degradation in process performance or process capability. However, in both cases a small level of initial impact was recorded, but was due to each project team's learning curve in coming to grips with the modified processes.

There are a number of general principles that can be drawn from the pilot results that ensure that process change has a minimal impact on project teams. The first is that all teams must be trained in the process change immediately prior to using it in a real project. The second is that process changes need to be introduced progressively in order not to overload the project teams with change. For example, if we had chosen to introduce both PRINCE2 and RUP at the same time then both the project managers and the development team would have been experiencing changes not only in their specialist areas, but more importantly on the interface between the development, which is iterative and the management, which is waterfall. Therefore, planning which processes to introduce and in what order is very critical.

Once the PRINCE2 integration was completed an attempt was made to define a generalised approach for carrying out such integrations in the future, based on the following and applied to the RUP integration:

- Keep it simple
- Make it “do-able”
- Maintain separation of concerns
- Maintain flexibility
- Create synergy

It can be seen from the RUP pilot that some of these generalised principles do in fact cause some contradictions. For example, maintaining separation of concerns can clash with the create synergy principle. On the one hand keeping RUP as a totally separate and loosely coupled interface would of course be easier to integrate. However, the QFSD execution and support processes in a lot of cases are better
than RUP and heavily overlap. Also, the RUP overall model does need to be integrated “seamlessly” with the QFSD model and hence to create synergy a deeper level of integration will give a much better combined product.

However, the researcher does feel that diluting the generalised principles to fit a specific case such as RUP is not appropriate and that the principles stand as aspirations in order that the most optimum level of integration is achieved for industrial methodology integrations.

The proposed generalised approach for integration of future industry standard methodologies into the QFSD framework is as follows:

- **Step one:**
  - Decide on the level of integration that will provide the best level of process improvement
  - *Keep it simple:* Always try to preserve each methodology to the level that it is recognisable by the methodologies user community
  - *Make it “do-able”:* The resulting integrated processes must be easy to use and have a supporting introduction organisation
  - *Maintain separation of concerns:* Where possible the methodologies should not overlap, unless there is a clear benefit in doing so.
  - *Maintain flexibility:* Integration of any methodology should not prevent the further integration of other methodologies
  - *Create synergy:* Where there is a deep level of integration required between methodologies this should provide benefits to both the QFSD framework and the integrated methodology

- **Step two:**
  - Compare and integrate the new process model in to QFSD framework. Using the following criteria:
    - Terminology: ensure that there are no misunderstandings between artefact naming in the QFSD framework and integrated processes
Guiding principles: ensure that the integration of the new methodology is easy to understand and to use

Roles: make sure that roles are clearly understood and to eliminate any duplicate or overlapping roles between the methodologies

Deliverables, products and artefacts: review, adopt or merge where appropriate the QFSD framework artefacts

Tools and techniques: Assess if there is a key benefit in adopting the new methodology's tool-set, if not then aim to integrate the methodology's artefacts in to the common central QFSD repository

- Step three:
  - Provide timely training to the project teams in the QFSD extended processes
  - Apply a standard set of key performance and capability metrics in order to measure the improvement or otherwise resulting from the use of the extended methodology in a pilot project

The three steps that form the generalised approach for industry standard methodologies integration into the QFSD framework should be applicable to the integration of any of the software development related methodologies. However, many of the elements within the three steps are equally applicable to the adoption of any new software development methodology.
11 Chapter 11 - Conclusions and further research

11.1 Introduction

This chapter reviews the aims and objectives of the thesis and will suggest that the QFSD framework is immediately useable by software practitioners as a practical software delivery tool-set. The literature review indicated that today's quality standards do not provide sufficiently detailed definition and support to enable software practitioners to achieve the goals and objectives defined within these standards, rather what is required is a pragmatic and practical software development framework. It is suggested that the QFSD framework, as a result of this research, meets this requirement.

In addition, Appendix G includes details on areas of case study that were not fully successful, how the approach was changed in some cases and lessons learned factored in to the QFSD framework case study approach. However, by careful selection of case study projects and preparation the majority of case studies were successfully executed.

During the thesis a number of new ideas have been developed that are candidates for further research and these will also be outlined.

11.2 Achieving the objectives

Please refer to Section 1.3 for the list of research objectives.

The research objectives were addressed by using a two phase approach. The first phase covered objectives 1 and 2, plus a sub-set of 5. This resulted in the development of the core QFSD Framework and associated online repository tool, validating it against real software development projects and consolidating the framework such that it was fully documented and could be deployed easily. The second phase addressed objectives 3, 4 and the remainder of 5. The framework was introduced and validated by applying it in a commercial software development environment on projects with increasing complexity. A number of these software development projects were selected as case studies. In addition, the business and IT
staff in the commercial environment had little or no previous experience of working with quality standards, or any professional software engineering methodologies. Hence in this case the framework was used as the basis for the introduction of the entire quality environment.

The conclusions reached by the research work will be discussed in the following sections against each of the thesis objectives.

11.2.1 Objective One:
The literature review in chapter two, supported by the researcher’s eighteen years experience in software development and project management, indicates the need for an integrated and prescriptive software development framework. Whilst there are many good software development methodologies and supporting tool-sets available covering project management and development, they tend to focus on a specialist process area. Those that try to cover the whole development lifecycle, such as ISO 9000-3 TickiT (1994) tend to be assessment frameworks rather than practical aids to the process of software delivery.

A software development framework not only needs to provide its own ‘core’ set of processes, it must also integrate with other industry standard methodologies and tool-sets in order to provide a flexible and optimised solution for any given development environment. However, having good software development processes coupled with strong specialist methodologies may not be sufficient. In order to be successful, the framework must be adopted fully and this means establishing a fundamental lifecycle model. This was a major lesson learned when introducing the framework in Celesio. The lifecycle model is tailored to the specific organisation and will optimise the application of the framework; refer to Chapter 9 for an example of how this was carried out in Celesio AG. This was a key lesson learned as the framework had a lifecycle model, but involving the development department in defining their own model strengthened both the framework and its adoption.

The literature review was carried out by reviewing published work including an analysis of a number of software project management, development and assessment
methodologies. The literature review concluded that methodologies fell into three basic categories, which are: assessment frameworks, iterative or rapid development methodologies and heavy-artefact, prescriptive methodologies that seem to be more suited to high investment military software development than to commercial development. None of the methodologies provided a holistic framework for establishing an adaptable and continuously optimised software development framework.

During the literature review the majority of the design requirements for the software development framework were established and used as the basis of the final design principles for the QFSD framework as detailed in Chapter 4. Chapter 4 also emphasised the impact of using specialist quality methodologies without considering the wider aspects of organisational participation, such as business case, stakeholder management and overall development lifecycle. This was reinforced by the first case study carried out at Fisher-Rosemount (Fisher-Rosemount Case Study One), which shows that by simply applying a specialist methodology the results can be poor impacting cost, budget and quality. As already described in Chapter 4, White and Fortune (2002) carried out a survey that showed over half of the companies who responded were forced to develop their own methodologies in order to address the shortcomings of existing methodologies.

Therefore, Chapters 3 and 4 establish the need and design principles for the development of the QFSD framework.

11.2.2 Objective Two:

The QFSD framework principles were defined in Chapter 4 as the initial starting point for the design. However, the framework was implemented iteratively over a number of years, based on multiple case study results, in order to progressively and thoroughly validate it against ever more complex software development projects and to refine and extend the processes within the framework.

Chapter 7 provides a complete test of all ‘core’ QFSD framework processes, by applying it as part of a detailed case study (Fisher-Rosemount Case Study Three) to
a large, new software product development project. Whilst the framework had already been used extensively at Fisher-Rosemount for enhancement projects, using it on a new development would exercise all processes within the framework. In addition, this would also provide a realistic comparison with the first Fisher-Rosemount Case Study One, which involved the development of the previous generation of the product, when it was also a new development.

This third case study showed how the ‘core’ framework could be successfully applied and, in fact, brought the development in on schedule, within budget and at a high quality level. Other notable results were that the project went smoothly, meeting the users needs and being executed in a controlled way, with no last minute panics and drafting in of excessive numbers of additional staff as had been witnessed in the first case study. The ‘core’ QFSD framework was fully adopted by Fisher-Rosemount, with a notable achievement of helping them to attain ISO 9000-3 TickIT (1994) accreditation. However, it must be understood that Fisher-Rosemount was a company producing technical software and hardware products and was used to applying quality standards and hence the staff appreciated the need for applying a level of formal control to software development.

A key enabler in the success of the framework is the fact that it is available as a central repository and accessible both using local clients in its first release and later via an internet browser. The introduction of the online repository was iterative in the same way as the framework processes themselves and, in its initial form, supported only the ‘core’ QFSD framework processes. The repository was necessary in order to meet the 9th design principle, which states "Framework must provide all framework artefacts to all management and staff who have a stake in the programme or project, in such a way that the artefacts are easily accessible, logically structured and under version control. The artefacts must be accessible from any managers or staff member’s desktop, development workstation or remote access device."

The initial repository was implemented on a Microsoft hardware and software platform, using the benefits of their internet browser, tree navigation technique and the Microsoft SourceSafe source code management repository, which also supports
all other document types. This provided the ideal environment for rapid development and deployment of the platform on to every managers and developers desktop (physical or virtual).

In order to support the introduction of the framework, and, specifically, the second Fisher-Rosemount case study, a basic online QFSD repository was introduced, which acted as the central repository for all process and project artefacts, providing the key advantages of:

- A common central configuration managed repository for all QFSD process and project artefacts
- Easy to access and detailed, online, interactive procedures and processes
- Structured project documentation management including approval workflow
- Structured software product design management
- Central online team working sites
- A central integration hub for the integration of all third party project tools such as test applications, defect management databases etc

The use of a central online repository proved to be a key success driver in the introduction of the QFSD framework. By providing a common point for all process guidelines ensured that all stakeholders and team members always had access to the correct procedures at the right revision level. All project specific artefacts were located in a single and well understood repository structure, which ensured that the members of each team had access to all the latest and relevant project information. All requirements, test cases, test results, defect analysis tools, and workflow tools were accessible from the common online repository. In fact the repository home page was used as a portal for accessing all artefacts associated with software development processes and project specific artefact deliverables.

Whilst the case studies were successful, it was clear that the projects involved had to be carefully selected in order to make sure they had the correct staff levels and skills, and no undue political constraints in the form of unrealistic cost and schedule
requirements. In addition, the project teams needed to also embrace the framework, which required it to become part of their everyday development approach. Furthermore, involving the key project managers and team leads in peer reviews at the end of each case study generated real support and enthusiasm for the framework which should be considered as a major contributory factor in its successful adoption.

11.2.3 Objective Three:

Scalability and applicability testing in this context covered the following elements:

- Ability of the framework to cover many different types of software project e.g. different required quality levels and complexities, such as programmes that contain multiple dependant projects, change requests, etc.
- Ability to cope with an environment of continuous business change and, hence, rapid delivery requirements
- Ability to operate with a business customer base who were not used to providing formal requirements, carrying out user testing or identifying business benefits, risks, etc.
- Ability for the QFSD framework to provide a central repository, which would handle many users and provide stable performance with an ever increasing database of current and historic project data.

Given the business and IT environment at Fisher-Rosemount, which was already familiar with the need for good quality IT processes, the introduction of the framework into a commercial company was a stringent test. The framework was introduced into a newly merged company consisting of AHH Pharmaceuticals (Wholesaler) and Lloyds Pharmacy (Retailer). These two companies not only had completely different approaches to software development, but also, as a result of the merger, had many duplicate and undocumented applications.

The introduction of the QFSD framework into this new company (Celesio AG) was successfully achieved by using a phased approach over a period of four years. The phases used were as follows:
• **Phase one:** Introduction of emergency processes in order to stabilise the new IT organisation and maintain and improve the current level of services to the business

• **Phase two:** Establish the quality framework tailoring requirements together with the new IT organisation to enable the introduction of the core QFSD framework

• **Phase three:** Introduce industry standard methodologies and extend the QFSD framework coverage

• **Phase four:** Extend the QFSD framework in order to support major business change programmes, both locally and across multiple sites

For this objective, this section concentrates on phases one and two only as they relate directly to the scalability and applicability objective.

The application of phase one would not be necessary in every introduction of the framework. The phase was required in order to quickly introduce a basic stable IT environment, educate the existing IT staff in good IT processes and to bring an appreciation to the business community of their role in the successful delivery of software solutions. Having completed this introductory phase, it was observed that the speed of introduction of the framework depends on the current appreciation of quality IT processes that already exist in a company.

Phase one also resulted in an unplanned benefit. By the introduction of emergency IT processes, taken from the framework, a minimum fundamental process set was identified that must either be in place, or form part of the initial framework introduction. Given that the newly merged company was essentially a green field site, the assignment of responsibilities for each IT process proved to be essential. This led to the further conclusion that a key part of the successful introduction of a quality framework is to ensure that the organisation itself is modified in order to allow it to gain maximum benefit from new IT processes.

Phase two included a reorganisation of the IT department to ensure that the framework processes could be introduced correctly and be easily enhanced.
However, since this was a green field site, no recognisable IT life-cycle model existed. Therefore, a new life-cycle model was introduced, based on that provided by the framework, but tuned to the organisation by involving both business and IT staff in the tailoring process. The result was a common alignment and understanding between business and IT of the software delivery processes. Again, the value of this step was emphasised and factored into the introductory approach for the framework with the addition of a new artefact called the Product Development Procedure (PDP). The PDP forms the top-level quality document covering all aspects of software delivery and records the agreed software development life cycle.

The first two phases indicated that the QFSD framework was sufficiently flexible in terms of addressing business and IT environments with radically different approached to quality. The introduction in to Fisher-Rosemount, which already had a basic level of IT solution development quality IT processes, enabled the QFSD framework to be adopted fully and quickly. This adoption provided a rapid improvement in quality IT processes and achievement of ISO accreditation. The introduction in to the newly merged businesses of Celesio AG presented a significant challenge, as there were a number of existing bad practices and a level of resistance to the introduction of quality control in to the development and management processes. However, the QFSD framework did improve the quality level in two years to a level 3 organisation based on the CMM guidelines (Caputo, 1998) and measured using the framework's Process Performance and Process Capability tool.

The framework also faced a number of additional challenges, for example, multiple business areas such as retail pharmacies, logistics, hospital business, services to pharmaceutical suppliers and management of multiple concurrent projects. The projects varied from intranet portals to the development of full ERP solutions. As can be seen in Chapter 9 the QFSD framework proved able to handle the many types of software projects and rapid development timescales, whilst giving improved predictability of project delivery to the business together with a consistently high level of quality.

The online QFSD repository proved to be scalable in handling the increased volumes
of artefacts and increased numbers of concurrent users. The repository became the basis upon which the company's project office was established and provided the authoritative source for all project documentation and governance monitoring.

11.2.4 Objective Four:

Within this objective the third phase of the framework's introduction into Celesio AG is considered. This will cover how the framework was adapted in order to integrate with other industry standard methodologies. The fourth phase, whilst showing good results, has been added to section 11.3. Future Research Areas, as the researcher feels at this point that further quantitative data is required before the final results can be published.

During the first two introduction phases of the QFSD framework it became apparent that the framework would need to allow integration with other methodologies. In the case of Celesio AG, there was a strong requirement to integrate with Prince2 (The Stationary Office, 2002) due to the UK business having to comply with this methodology when delivering healthcare solutions to the NHS. In addition, certain European business units within the Celesio AG group companies were in the process of adopting sub-sets of RUP (2007). The researcher identified two further challenges that could be met by addressing this requirement. Firstly, that the QFSD framework would be applicable to companies that already had 'islands' of quality methodologies in place, or who adopted the framework, but were required, due to external pressure, to use a particular methodology. Secondly, that the integration exercise would provide a solution to the problem of integrating methodologies that have fundamentally different life-cycle models. In this case PRINCE2 has a waterfall model, whilst RUP has an iterative model.

Chapter 10 describes the approach used in order to prove the integration of both methodologies by applying the integrated IT processes to four real software development projects at Celesio AG. The results and metrics used in order to measure the success of the integration are also described in Chapter 10. Apart from the creation of generalised interfaces for merging data repositories, automatically adding standard artefact attributes used in the QFSD framework and general import
utilities, the initial approach was to treat the integration of each new methodology in a unique way. However, once the integration of PRINCE2 was completed, an initial generalised approach started to form.

The first attempt at integrating PRINCE2 was not very successful as this tried to completely overlay the corresponding QFSD framework artefacts and processes in the Organisational and Management process groups. The resulting integration would require a significant amount of reconfiguration of the framework and would leave inconsistencies in the existing framework processes when compared with the overwritten PRINCE2 replacements. Further, if the integration approach continued to treat each new methodology as a special case then there would be inconsistencies in the integration approach for each methodology.

This suggested that a generalised integration approach should be developed, which optimised the use of the QFSD framework processes, reduced time and effort for the integration and could be applied to many types of methodology. Although the actual results provided address only two methodologies, they are representative in terms of their differing life-cycle models.

The generalised integration approach is described in the conclusions section of Chapter 10. However, in summary the generalised approach is as follows:

- **Step 1:** Ensure that an appropriate level of integration is planned, such that maximum benefits are obtained from each methodology and that the integrated methodology is recognisable by the methodologies user community

- **Step 2:** Execute the integration on the following levels and ensure that a common artefact repository is maintained:
  - Terminology
  - Guiding principles
  - User Roles
  - Artefact deliverables
  - Tools & techniques
• Step 3: Ensure that project teams are trained correctly in a timely fashion before using the new methodology. Also, establish a set of key performance and capability metrics in order to measure the improvement.

The generalised approach will be used for the integration of an SOA repository as the next step. Information concerning this integration can be found in section 11.3.3.

11.2.5 Objective Five:

Chapter 6 describes the structure and development of the QFSD online repository. The repository has been used during the development and validation of the framework as described throughout this thesis. The concept of an online central repository has proved crucial in the introduction of the QFSD framework, both in terms of integration of the processes with project management and development teams and in delivering the quality benefits that are naturally delivered from having all artefacts controlled in a central repository.

The effectiveness of the repository has been described in a number of chapters. However, it is worth mentioning that the repository platform itself has also been evolved. Initially the repository was a simple graphical interface that accessed the Microsoft SourceSafe database. Microsoft then published information concerning each function within the SourceSafe database and this was used in order to build a functionally rich graphical interface. The next step was to add a workflow tool to the front end in order to provide process workflows for document review and release.

Open interfaces were subsequently added to enable other third party tools to be integrated with the repository and finally the ability to interface to other document management repositories was added.

In terms of meeting the objective, Section 6.2.5 provides the customer views and endorsements concerning the use of the repository.

11.3 Future research areas

The following sections provide a short summary of areas for future research and indicate where work is already in progress.
11.3.1 Application of QFSD to European programmes involving business change

Phase 4 concerning the introduction and validation of the QFSD framework in Celesio AG involves the validation of the framework for large international programmes. In effect phase 4, is still in progress. An international programme in the context of Celesio AG includes both business process and IT changes across two or more business units in different European companies. The QFSD repository is web based and, therefore, accessibility across multiple countries over a suitable network. In the case of Celesio AG this is a European wide MPLS network over which access to the repository has been shown to work.

The management processes within the framework are derived from the CCTA programme management standard (CCTA, 1999) and hence the framework has the basic capability of managing multiple projects within a programme. However, in terms of tracking and reporting across programmes, further work is required in terms of cumulative reports showing program level progress and costs.

The QFSD framework is being used on a major European business process change project. Once the project is completed, a review of the QFSD framework will be carried out and any recommendations will be factored in to the next release. This process will be repeated for a further two major programmes before the QFSD framework can be deemed as being validated for use on major European business process change projects.

11.3.2 Integration of Cassandra into the framework

The researcher has responsibility for the overall Celesio AG IT Strategy and for ensuring that each of the 36 business units across Europe has business aligned IT Strategies. As part of the business alignment approach definition, which includes the creation of an agreed Enterprise Architecture and agreement on the Operating Model classification of each business as defined by MIT Sloan Center for Information Systems Research and described by Ross et al. (2006), it became evident that there is still a gap between the businesses and their various IT departments. The key
indicator for this observation emerged when the researcher carried out an analysis of
the actual benefits delivered by the major projects delivered in each of the 36
business units during 2007. Those business units that were using the QFSD
framework, which promotes the concept of defining an agreed business case,
showed a good correlation between expected and delivered business benefits.
However, those business units not using the QFSD framework were much less
successful in obtaining business benefits and certainly could not really measure them
following the deployment of the software solution.

The lack of clarity and measurement of business benefits delivery would seem to be
a general trend according to Edwards et al (2007). Their paper indicates that a good
proportion of project failures are not as a result of new technology or poor project
execution, but, rather, problems associated with the scoping of the project, creation
of a real business case and the approval process applied to the project. These
problems identified as the result of a survey of a hundred IT managers were further
classified as:

• Lack of a business case process
• Very poor cost/benefit analysis
• Failure to identify associated business process changes and impacts
• Poor project approval forums
• Approval groups lack of understanding of the business case

This points to a communication issue between the business and IT teams. The
logical solution is to create a common language between the business and IT, plus
the adoption of a culture that only approves those IT projects that have real and
measurable benefits. To create this common language the researcher introduced in
to Celesio AG a graphical project definition process (a.k.a Cassandra) from the
Cranfield School of Business Management. Figure 11.1 provides an indication of the
graphical nature of this approach.
In brief, the Cassandra method provides a visual template that is used by IT in the initial discussions with the business. The template is usually put on a white board and is populated by a combined business and IT team. The dependency diagram is completed from right (Vision) to left (Infrastructure). The beauty of this approach is that it forces the business to define real benefits and to understand what the impact on their existing and future business processes will be and the changes they will have to undertake in each business area or department in order to realise those benefits. Basically, the method pushes the business to be proactive rather than simply requesting an IT solution. This is emphasised in the method by having the IT elements as the last elements to be completed.

Early results of using the method are very promising. The most dramatic result so far can be found in the request from a full-line wholesale business in Celesio Portugal for a Customer Relationship Management IT solution. A Cassandra analysis was carried out with the business sponsors and IT management, which resulted in a realisation that out of 30 identified benefits, the top three, which covered 80% of the overall
benefit, could be obtained from their existing applications. The beauty of the method is that the linked graphic enables benefits to be traced through the analysis showing what would be required in terms of business and IT effort verses the value of the resulting benefit.

The next step, in terms of further research, is to gather a wider set of results before starting integration with the QFSD framework. The integration would introduce this as part of the business case process, which is grouped under Organisational Processes in the framework. Basically, attributes would be added behind each graphical element for automatic population of a basic business case. The attributes would be tailored by the position of the element in the graphic. For example elements would include attributes such as description, effort, cost, schedule, risk etc, but the critical attribute is ‘owner’. All benefits and their delivery need a business owner. The business case is a major contributor to successful delivery of software projects as asserted by the researcher in a published paper (O'Neill and Dawson, 1999). Providing a graphical tool for the creation of a business case would be a major additional tool within the QFSD framework. The design of the graphical interface is in progress.

11.3.3 Integration with an SOA service repository

A major part of the researcher’s role at Celesio is the definition and delivery of standard application architecture across all European businesses. In terms of technologies this is actually achieved using different approaches depending on the actual business division, IT investment budget and IT organisation’s maturity. The maturity level of an IT organisation within Celesio is determined by its level of adherence to the Celesio IT Governance Framework (2008). However, the standard application architecture is based on standard Service Orientated Architecture (SOA) principles. In order to support the introduction and management of an SOA in Celesio AG, further research work will be carried out with the target of integrating the QFSD framework with a Service Oriented Repository. At this point the decision has not been taken as to the general approach. There are two options under consideration. The first is to purchase a commercial service repository product and
provide an interface using the rules as defined in Section 10.4. The second option is to develop a service repository as an extension to the QFSD document repository.

Whilst a decision has not been taken concerning which option to take, the requirements for the service repository are in the process of being defined, working with a number of IT departments within the Celesio AG group companies. The following sections describe the basic requirements for the service repository based on a series of workshops with each of the IT departments. Where appropriate, the requirements are extended to include further descriptions in order that those unfamiliar with SOA can attain a clear understanding. It is assumed that the reader has a very basic understanding of Web Services and Enterprise Service Bus concepts.

In order to attain the benefits of SOA the following governance areas must be introduced:

- Service Lifecycle
- Service Repositories and Registries
- Configuration Management and Version Control
- Run-Time Management of Services

High level requirements on each governance area are provided in the following sections. Governance in this context provides a framework within which the requirements for design, deployment and reuse of services are controlled. A successful SOA requires certain prerequisites to be in place that are described as follows:

**Business Ownership**

For an SOA to be effective, it is critical to get business ownership of services and support for the service-oriented approach.

It is clear from industry case studies (SOA consortium, 2007) that the adoption of an SOA realises the most benefit when the business and IT parts of the organisation are
aligned. This implies a good understanding of the business processes, the drivers for change and the current IT landscape.

**Architecture Framework and Policies**

It is critical to adopt a set of architectural standards and policies (Sprott and Wilkes, 2004) that control the development and deployment of services.

It is important to develop a service portfolio. Where possible produce a top-down view of the business services required; where this is not possible, develop services on a project by project basis but still try to build the overview.

**Asset Management**

Services and related artefacts are assets of the organisation, and to achieve effective reuse they need to be actively managed. Recognition of this leads to the use of a service repository to store information about services that can be accessed by developers, project managers, architects, business analysts and others interested in the development of and reuse of services.

**Lifecycle Management**

The lifecycle for services is different from that of other software artefacts because of the way in which services are designed to be reused. Whereas most software has a lifecycle in relation to a single project in which it is developed, the service lifecycle cuts across multiple projects. Because of this, it is important to be clear about the roles and responsibilities in relation to requesting and approving reuse of services and the quality checks that are applied to services and the steps in the lifecycle.

**11.3.3.1 Service Lifecycle**

Services are developed according to a clearly defined lifecycle. The service lifecycle defines the activities in the life of a service from initial conception through to deployment and subsequent withdrawal from service.

Figure 11.2 shows the lifecycle from conception to deployment as defined for Celesio AG.
Figure: 11.2 Service Lifecycle
This lifecycle diagram, which focuses on the development part of the lifecycle shows that there are a number of points during the lifecycle at which it is important to check issues such as whether a service already exists, whether it can be reused and whether a new service fits into the overall architecture. The lifecycle also has a set of defined roles and responsibilities which should be configured in a service repository.

**Service Repositories and Registries**

The fact that it is possible to build an SOA in which services can be located dynamically and bound to at run-time is an important characteristic of SOAs and one of the selling points of the approach (Sprott and Wilkes, 2004). The standard approach in web services is to use a UDDI (Universal Description, Discovery and Integration) Registry although there are other technologies that support this dynamic binding, such as Sun’s JINI. However, it is not essential to be able to locate services at run-time. There is an overhead associated with looking up a service in a UDDI registry, retrieving its WSDL (Web Service Definition Language) and binding to it dynamically. To achieve reuse of services, it is more important to maintain details of services in a service repository.

**Repository in the Lifecycle**

Figure 11.3 establishes the SOA repository in the context of an organisation developing services in an SOA environment and defines the relationships the repository has with other SOA-supporting tools. It shows the development of a service from its initial proposal and specification through to a deployment of an actual instance of that service.
When a service is proposed, its specification is lodged in the repository. This will allow potential users of the service to identify the service development and register an interest in eventually consuming the service. It will also help reduce duplicated development effort.

During the development of a service, the service-related artefacts, such as source code and documentation, are created with the aid of the normal configuration management systems including source code control applications such as ClearCase™, CVS or SourceSafe™.

When the service implementation is complete, an asset should be created in the repository to describe the implementation, and the artefacts relating to the implementation, such as WSDL should be made available from the configuration management system. These artefacts should be copied or made available by reference to the configuration management system by use of a URL. For example, WebCVS provides a suitable front-end for CVS, where artefacts may be referred to via a baseline label within a CVS repository.
When an implementation of a service is deployed it becomes a service instance. Typically the executable artefacts for the service implementation will be deployed into the run-time environment. An asset to describe this instance should be created in the repository along with appropriate artefacts. Finally, the service is published in a UDDI registry to allow applications that require UDDI support to discover it. This last step should also be automated and must support automatic publication into several vendors' UDDI registry products including jUDDI and Systinet registry.

Figure 11.4 shows a runtime view of a typical SOA environment indicating other systems that are used for managing deployed services. These include an optional UDDI registry for run-time discovery of services and service monitoring tools.

![Diagram showing runtime context with UDDI Registry](image_url)

**Use a UDDI registry if necessary**

A UDDI registry may be introduced into the run-time environment in order to support the standard approach to locating services. If this is done, then the process of publishing to the registry should be automated using the repository.

**Use an SOA repository to achieve reuse and enforce policies**
An SOA repository is essential to achieve the reuse of services. Without a standard location in which to store information about candidate and actual services, it is difficult to see how projects will be able to benefit from the reuse of existing services.

The repository should support the enforcement of the service lifecycle by means of an approval process. It should be possible to configure the repository so that before a service can be published, it must be approved by someone in one or more roles.

The repository should also be used to enforce development policies by requiring that certain artefacts have been placed in the repository and by checking the validity of WSDL documents.

11.3.4 Configuration Management and Version Control

A service repository will not provide the same functionality as a configuration management system. The repository should be integrated with the underlying QFSD repository configuration management system by creating links to the repository through a web-based front-end.

There are two aspects of configuration management and version control that need to be addressed. The first relates to the service development lifecycle, the second to the management of multiple versions of services in deployment.

Configuration management in the service development lifecycle

In order to achieve reuse of services, it is necessary to manage the status of services throughout their lifecycle. It should be possible for potential consumers of services to find details of those services in the repository. The information about services must include information about their status. There is no point in attempting to consume a service that is deprecated or withdrawn.

From a reuse perspective it is important to know also what services are being considered: a project may have an idea for a service but not have developed it yet. If another project plans the same or a similar service, then it should be possible to find out that the first project has already considered it and possibly begun to specify it. This makes it possible to avoid duplicated effort in developing services, and to ensure
that services that are developed meet the needs of as many potential consumers as possible.

The next step in the research is to generate a formal set of requirements for the SOA repository and evaluate market available products against the option to extend the QFSD framework.

11.4 Overall Conclusion

This thesis has shown that a more pragmatic and practical software development framework can be developed, which provides software practitioners with an immediately useable holistic software development tool-kit. The Quality Framework for Software Development (QFSD) has been used successfully on a significant number and types of projects and adopted in two quite different companies. Both companies showed a marked improvement in software delivery times, costs and quality. In the case of Celesio AG, the framework rescued the UK business units IT capability following acquisition and provided the foundations for their future IT department.

From an experience and understanding point of view, the researcher has observed that development teams take the view that imposed quality standards are an unnecessary overhead, that project management methodologies are for project managers and that their concern is limited to code related development tools. What the framework design does is integrate project, quality and development management and associated processes in order to enable a seamless overall development capability. The major surprise was that following the initial case studies each of the previously separate factions: Marketing, Project Management and Developers started to operate as a single team with better communications and an appreciation of each team's contribution to the success of the software delivery. It could be argued that this was the influence of the teams being briefed on the various case studies, or better inter-team communications which was a result of the case study process, or indeed, the improved framework processes and underlying common repository. It is the researcher's view that all these were contributory factors to the frameworks success. In order to be successful, any framework needs to be
accepted and supported by the teams adopting it and be seen to improve their day-to-day activities.

The research approach was based on a pragmatic philosophy and, as such, the delivered framework is not a theoretic model, but a practical framework, used in real commercial IT projects and available for use by software practitioners in the software industry. In addition, the framework and future research areas provide a platform upon which further academic research can be undertaken.
12 References

References:


Boaden R.J., (1997) Total Quality Management, Volume 8, Number 4, 1 August 1997, Pages 153 to 171. Publisher: Routledge, part of the Taylor & Francis Group


Boland, R.J. JR. (1979) Control, causality and information systems requirements, Account. Org. Soc. 4, 4, Pages 259 to 272.


Capability Maturity Model, Version 1.1, Feb 93, CMU/SEI-93-TR-25


Consultative, Objective and Bi-Functional Risk Analysis. (2005) www.securitypolicy.co.uk/riskanalysis (accessed 07.05.05)


Gopelrud, E. (1989). Presentation at Demand Reduction Task Force meeting, Prevention/Education Programs of ADAMHA.


Liebs, S. (1992). We're All In This Together Information Week, (Oct 26), Page 8.


Peirce, C.S. (1878) How to make our ideas clear, Popular Science Monthly, reprinted in Thayer


Wasserman, G.S,. (1993) On how to prioritise design requirements during the QFD planning process", IIE Transactions 25(3), Pages 59 to 65.


(Volume 29, No 11) Pages 69 to 77.


Yin, R. (1994). Case study research: Design and methods (2nd ed.). Beverly Hills,
CA: Sage Publications.

Yin, R. (1989) Case study research: Design and Methods, Sage Publications, Beverly
Hills, California.

Publishing.

Sage Publishing.

Success, Addison Wesley Longman, Harlow, UK.

Zarrella, P. F. (1990) Case Tool Integration and Standardisation, SEI Carnegie
Mellon University.

Zeineddine, R. (2005) SQIMSO: Quality improvements for small software


Pages 43 to 44.
13 Acknowledgements

- Extreme Programming (XP)
- Scrum™
- Dynamic Systems Development Method™ (DSDM)
- Crystal™
- Adaptive Development™
- PowerBuilder™ is a trade mark of Sybase Inc.
- SQCpack™ is a trademark of PQ Systems.
- Microsoft Project™ is a trademark of the Microsoft Corporation.
- Microsoft Exchange™ is a trademark of the Microsoft Corporation
- Microsoft PowerPoint™ is a trademark of the Microsoft Corporation
The following shows a chronological history of the decision point in the first case study taken from the lesions learned records of that project.

Table A1 - A chronological history of the first case study

<table>
<thead>
<tr>
<th>Date</th>
<th>Project Decision Points</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jun 87</td>
<td>Concept for the new configuration product.</td>
<td>Concept for a relational database version of the existing ASCII based configuration product was first discussed. The company desired database was Oracle, due to its wide use in the parent company. A relational database was thought the best way forward. Using the benefit of hindsight, selection of this technology was correct. However, expertise in relational technology would prove difficult to find.</td>
</tr>
<tr>
<td>Oct 87</td>
<td>Marketing Product Manager Appointed.</td>
<td>At this point there were no written requirements, just a description for budget purposes. This situation was to continue until late in the project.</td>
</tr>
</tbody>
</table>
| Dec 87 | Development Project Manager Appointed. | • Investigation in to which relational database to use. The selection of the database vendor was questioned and defended throughout the project.  
• Still no Marketing requirements specification. |
<table>
<thead>
<tr>
<th>Date</th>
<th>Project Decision Points</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feb 88</td>
<td>Project Manager assigned to another project.</td>
<td>This was again a result of lack of marketing direction. The project was to evaluate a PC front end for the configuration product. However, this was a diversion as PCs at that time did not have the horse power required. This wasted around two months of the project.</td>
</tr>
<tr>
<td>Mar 88</td>
<td>Draft Marketing Specification released.</td>
<td>This was written in a hurry and had not really been validated with customers. The Product Manager was then diverted off this main project for a few months.</td>
</tr>
<tr>
<td>Apr 88</td>
<td>First set of project estimates produced.</td>
<td>These estimates showed a timeline of 15 months, 9 developers i.e. 135mm. These estimates were plucked from the air by an inexperienced project manager who had not managed a large software project before, coupled with the lack of historical data from previous projects. The actual project took 24 months and required 595 man-months.</td>
</tr>
<tr>
<td>May 88</td>
<td>Detailed evaluation of relational database vendor products.</td>
<td>This took a month when there was really only one product that was suitable for the application. A good deal of this evaluation was necessary to fight off political pressure to select the corporate standard Oracle. This decision was defended by the project manager throughout the project. Selection of technology should be left to the technical experts and not second guessed throughout the project.</td>
</tr>
<tr>
<td>Date</td>
<td>Project Decision Points</td>
<td>Commentary</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------</td>
<td>------------</td>
</tr>
</tbody>
</table>
| June 88 | • Product & Project Managers visit a number of customer sites to gather requirements.  
• Sybase verses Oracle decision questioned again.  
• Schedule for the project was set by senior management. | • Six months in to the project and only just capturing the requirements. These ‘requirements’ were then transferred from the Project Manager to the development team, with no further Marketing involvement. This transfer of requirements / concept development took around four months of the Project Managers time, thus taking him away from the more direct Project Management responsibilities. A lot of time was lost during this stage of the project due to a lack of marketing support.  
• Senior management set the release date as June 89. This was a year from this date and tree months less than the project manager's original estimate of 15 months. |
| Sept 88 | New project estimates based on developers understanding of requirements. | The new estimates were 178 man-months development with 2.75 man-months of testing. The testing effort estimates again shows a complete lack of appreciation of testing large software systems. |
| Oct 88 | • Marketing requirements specification available.  
• Request made for additional resources. | • Eight months in to the project and no code cut.  
• The resources at this time numbered six. In the space of three months the resources increased to eighteen, peaking at 30 during development and 40 during release testing. The resource increase followed a typical
<table>
<thead>
<tr>
<th>Date</th>
<th>Project Decision Points</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feb 89</td>
<td>Marketing request that the product should run on a second platform.</td>
<td>Classic requirements creep, with large impact on release testing and the products installation.</td>
</tr>
<tr>
<td>Apr 89</td>
<td>• Requirements established.</td>
<td>• Team was working on code at this time.</td>
</tr>
<tr>
<td></td>
<td>• Release date moved to Dec 89.</td>
<td>• Release date starting to move as size of project starts to be understood. Date slipped from June 89 to April 90.</td>
</tr>
<tr>
<td></td>
<td>• Beta sites selected.</td>
<td>• Only four beta sites were selected. Given the performance of typical beta sites, a greater number of sites would be required to test performance and installation (the two major customer immediate product perception issues).</td>
</tr>
<tr>
<td>Jun 89</td>
<td>Release test planning starts</td>
<td>This was a little late in the process.</td>
</tr>
<tr>
<td>Sept 89</td>
<td>Product demonstrated at trade show.</td>
<td>Team effort diverted at a critical point in a project that was already late.</td>
</tr>
<tr>
<td>Nov 89</td>
<td>Products Beta Test Plan available.</td>
<td>The first objective stated at the start of the Beta Test Plan was &quot;DEVELOP DEFECT FREE SOFTWARE AND PRODUCT&quot;</td>
</tr>
<tr>
<td>Date</td>
<td>Project Decision Points</td>
<td>Commentary</td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>DOCUMENTATION&quot;. This reflects the Crosby culture influence.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>There were only 4 Beta Sites chosen. This is not nearly sufficient to flush out performance and installation problems.</td>
<td></td>
</tr>
<tr>
<td>Feb 90</td>
<td>Indications from Marketing that the ENVOX product was very slow.</td>
<td>This was largely ignored. The most surprising aspect was that marketing benchmarked the product themselves. This should have been carried out by the development team.</td>
</tr>
<tr>
<td>Apr 90</td>
<td>Strong written communication from Marketing on the unacceptability of the products performance. This was sent to all management levels.</td>
<td>Written communication from Marketing Product Manager, which stated that the product would have totally unacceptable performance for customers. This against a backdrop of it being a productivity tool, should have been a trigger to take immediate action. The development team predicted only a 20% performance improvement before release (based on current schedule time available). Marketing were also at fault as they agreed to the release, by accepting a promise of a quick second release P1.1 to fix the performance issues. This was a major release blunder from which the product never really recovered. The project should have gone for an extensive field trial release (got it to the field, whilst working on the performance improvements). This would have established just how unacceptable the performance was,</td>
</tr>
</tbody>
</table>
and generated some additional time for the problem to be addressed.

**May 90**  
Project team forced to start testing before the code was completed.  
The ENVOX project used a 9-week pre-product testing approach. What this meant was that the product was being tested whilst the code was still being written. This is not a good approach, unless the areas of code being tested are totally separate from those still under development. The result was that a large number of defects were generated which proved very difficult to fix, as the code area was still under development. This generated an abnormally high back log of defects, which were then difficult to repair. The inflated defect numbers were used as a weapon to attack the project team on the run up to release. A better approach would have been to use the defect / hr rate information, in order to measure stability. However, only started to record and fix defects once the code was stable.

**Jul 90**  
- Product Release Meeting.  
- Product is mass shipped.  
- The release meeting took place, with the following release criteria:  
  1. 'The product must be safe and useable'. This should have included the products Performance. It is ironic that the definition of useable used in the release meeting was
<table>
<thead>
<tr>
<th>Date</th>
<th>Project Decision Points</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>'Useable means that the customer will be satisfied with the overall product performance.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Migration of test configurations from the existing product, through to download (download to target devices over a data highway) shall take place with no defects.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. System release test configuration should be downloaded with no defects.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. No level 1 defects for 500 hrs testing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5. All level 2 defects to have a work-around.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6. Must be able to install the product on a VAX 3100. This did not include extensive customer installation testing. The installation was very complex. This was reported in early customer feedback.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7. Must be able to confirm that correct configurations have been downloaded to all target devices.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The release meeting held on the 12\textsuperscript{th} of July 90. The release notes contained a statement that prolonged use of the forms system (user interface) will cause the product to crash.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The product did not attain the goal of 500 test hours without a level 1 defect. The products best performance was 300 hours</td>
</tr>
<tr>
<td>Date</td>
<td>Project Decision Points</td>
<td>Commentary</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>without a level 1A defect.</td>
<td>• At the release meeting, the level 1 defects, where divided into level 1A &amp; 1B. This was clearly a ploy to gain support for the release.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The product was shipped on mass over period of two weeks (120+). This was half a typical year's shipment. This swamped the Technical Support Group with installation problems. This coupled with the fact that the product was new, with the new relational database technology, left the support group sadly unable to provide sufficient customer support.</td>
</tr>
<tr>
<td>Oct</td>
<td>Customer Negative Feedback Starts</td>
<td>• Majority of customers cite performance as the major issue.</td>
</tr>
<tr>
<td>90</td>
<td></td>
<td>• Technical Support group's loading is becoming significant.</td>
</tr>
</tbody>
</table>
|       |                                     | • As a result of the P1.0 release, the P1.1 bug fix release started to show signs of project management process execution improvement. The project was set up to complete some of the missing features, improve performance and fix more defects. It was not until P2.0 that performance was acceptable. P1.1 should have addressed performance only, which was the major.
<table>
<thead>
<tr>
<th>Date</th>
<th>Project Decision Points</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>issue.</td>
</tr>
<tr>
<td></td>
<td>8. Improve testing: regression + specific testing focus on new code areas + tech support involvement + technology developers to spend 2/3 weeks on site with operations / tech support personnel (sort of bolting the stable door after the horse has bolted)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9. Control release numbers to the field</td>
<td></td>
</tr>
</tbody>
</table>
15 Appendix B – Example of failure in new product development using waterfall methodology

The following is a summary of the independent IBM review (IBM Review, 2003) of the DCH software development.

**B1 Summary and recommendations:**

The IBM review has broadly confirmed the Corporate IT findings. The findings can be summarised as follows:

- The product develop by DCH and released in stages to Lloyds will not provide a long-term viable or supportable solution for the Lloyds Business in its current form. Lloyds have agreed this statement.

- Whilst DCH indicated that their standard development approach was based on the Rational Unified Process (RUP), it is clear now that they followed a standard waterfall methodology. This single decision resulted in the majority of the technical issues that caused this new product development to fail.

- It is clear that the programming standards, design understandings of modern software architectures are all weak in DCH.

- What amounts to a complete re-write will be necessary in order to release the product to the Lloyds estate.

- Performance with the current design of the NeDS product is still an issue requiring the shortfall to be met with additional hardware. Even then, the bandwidth issue is not fully resolved.

- Any re-write or significant re-engineering will require the Lloyds team to be strengthened in terms of both requirements management and software development planning, construction, testing and overall quality assurance capabilities.

However, we need to consider the impact that the DCH issue has on the Lloyds Business. Currently the Lloyds Business is under pressure from the NHS to provide more and more Healthcare Services.
One of the major NHS initiatives is that of e-scripts. In this case, Lloyds and Corporate IT have come up with an interim solution using the existing dispensing system ‘PMR'. None-the-less, the initiatives from the NHS are continuing to develop and in the majority of cases, will not be realisable in Lloyds existing ‘PMR' system. Therefore, the drivers are there for Lloyds to release a new Dispensing System to the market as soon as possible. Competitors in the UK market, such as Boots the Chemists, have failed to deliver a new dispensing product to the market. If Lloyds achieve a centrally based product, it will improve their position the UK market. Given that Lloyds has no choice but to move forward and select a new Dispensing System, the realistic options are limited.

One further issue to take into account is the suitability of the current NeDS product to the Lloyds Business. Remember that the current ‘PMR' product provides a simple and very quick user interface for dispensing. Even if the current NeDS product were operating to its design performance criteria, it would be a more complex and slower interface to that which the Lloyds pharmacists are used to working with today. Therefore, before embarking on any re-engineering or a total re-write, the user interface needs to be validated with the practising pharmacists.

The options available are:

1. Lloyds could search the market for a commercially available product. It is very unlikely that a product exists that will meeting Lloyds requirements.

2. Perform a complete re-write of the product using either a commercially available application server product or a standard JAVA framework such as ‘Strutts', etc., however, use the current product from DCH to do a two-store pilot as soon as possible. This would validate the operational design of the product before the re-write is started. This will ensure we are building the right product for the Lloyds business, both now and for the future. The two-store pilot would be based upon the current code line with only bugs fixed in order that the product provided basic functionality. It would be unstable and require a lot of in-store support. If this was done in-house, then both the Lloyds management and development teams would need to be strengthened by
experienced resources. Outsourcing could again be high-risk and gives morale issues in the existing Programme team. However, in terms of timescales and cost a tightly managed outsourcing would be the best option.

Option 2 is shown diagrammatically in Figure B1.

![Diagram showing Option 2]

**Figure B1: Celesio Option 2**

This is the preferred option of the Corporate IT team.
3. The preferred option of Lloyds is to bring the DCH product in-house and make significant changes to the existing code line in order to make the product stable and secure. The team’s estimates to complete this are between 600 and 1000 man days. This version would be released to 30/50 pilot stores. At that point, the decision to either continue rolling out beyond the 30/50 stores, or to stop and re-engineer, or to continue the rollout and re-engineer in parallel, would be made. Option 3 is shown diagrammatically in Figure B2.

Re-engineer Existing Code Line (V1)
- Release to 30/50 stores
- Stable
- Not fully performant
- Not scalable
- Reduced functionality

Continue to re-engineer Existing Code Line (V2)
- Release to 1350 stores
- Stable
- Performant
- Limited scalability
- Full functionality

Re-implementation (V3)
- Release to 30/50 stores
- Stable
- Not fully performant
- Not scalable
- Reduced functionality

Figure B2: Celesio Option 3.
This would be the most expensive option and leaves the re-implementation start date very late. The re-implementation would be delayed by 12 months. In terms of Lloyds business objectives, it would seem to get the product to the market sooner. However, this raises two major questions:

1. Given the state of the current product and its design, could Lloyds ever make it suitable to release to 1350 stores?
2. Could the Lloyds team complete the re-implementation to a standard and timescale acceptable to the Lloyds business?

B2 Recommendations

The recommendations are as follows:

- Transfer the product from DCH and move it in-house.
- Adopt a true iterative approach to the development in order to control development and reduce delivery risks.
- Re-organise the NeDS team in order that it has clear ownership and responsibilities for the development of NeDS. Specifically, strengthen the development team management, quality function and JAVA expertise.
- Make the necessary changes to the existing product such that it can be deployed in a technical pilot in two stores. (Option 2 as described earlier). The changes would not be as defined in order to make it robust and performant. Rather to get two stores using the product in order that its operational suitability can be determined. Lloyds store operations function should monitor the product's use in-store in order to validate its operational design.
- Once the operational suitability has been determined, this may need a re-design of the product. This re-design should be factored into the replacement product.
• Start a new project with a much-strengthened team to deliver a new product. This product would be designed with performance and flexibility as a major design criterion.
16 Appendix C – Core QFSD Processes Model

The following tables contain the core QFSD process definitions for the Organisational, Management, Support and Implementation components. Each process has a process description, a set of objectives, associated techniques to achieve the process objectives, online tools to support the process if applicable and background as to why the process is necessary.

The processes in Table C1 are part of the QFSD online repository screens and hence only a proportion of them have the full details provided in the following table. However, the table should be sufficient to provide a detailed understanding of the major processes and the level of guidance provided.

Table C1 - Core QFSD processes

<table>
<thead>
<tr>
<th>Process Description &amp; Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Organizational Processes:</strong></td>
</tr>
<tr>
<td>• Process: Company Objectives</td>
</tr>
</tbody>
</table>

Senior Management within the organization must ensure that they communicate the organization's business goals, and provide the necessary decision support throughout the duration of a software project. However, for the software groups within the organization, this needs to be translated into directly applicable terms and objectives. For example if the goal is to move the company's products to the Microsoft NT platform then that should be stated explicitly to the software groups.

• Objectives:

1. To ensure that the company has an efficient mechanism for reviewing the company's performance against its leading competitors in the market place.

2. To ensure that the company has an efficient mechanism for the collection of customer needs.

3. To ensure that the resources responsible for software product
**Process Description & Objectives**

<table>
<thead>
<tr>
<th>Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>development understand how the organizations business objectives impact their product developments.</td>
</tr>
<tr>
<td>4. To ensure that all software products are aligned with company objectives, thus reducing rework or not meeting the customer's needs.</td>
</tr>
<tr>
<td>5. To ensure that a product is released in the optimum 'market window'.</td>
</tr>
<tr>
<td>6. To enable Project Management to make better project decisions / development trade-offs.</td>
</tr>
</tbody>
</table>

- **Techniques**

1. Objective (1) can be achieved by having an independent group carry out a customer satisfaction survey of their products against leading competitors products. The advantages of having a third party carry out this survey on a periodic basis, is that the results are likely to be more representative and dispassionate, than if the company surveyed its own customers.

2. Objective (2) can be achieved by having a number of customer focus groups or steering committees, which meet periodically. In addition, a metric, which captures major customer issues and their resolution, is also useful in focusing the organization towards improved customer satisfaction.

3. Objective (3) can be achieved by communicating a clear set of company goals translated in to each area within the business; Development, Marketing, Manufacturing and Operations, must be communicated. This should be done in both documentation form and using monthly or quarterly senior management communication briefings to all staff.

4. Objective (4) can be achieved by making sure both Development / Marketing senior management are involved in the project approval stages. This involvement must include more than just a funding review of the projects. A formal review process involving all stakeholders is used. Refer
### Process Description & Objectives

to the rows below entitled "Executive Test Plan Review Process" & "Executive Release Readiness Review Process".

5. Objective (5) requires the company to have a good awareness, not only of its customers needs, but also of its competitor’s product strategies.

6. Objective (6) requires that Project Management have access to Senior Development and Marketing Managers, in order that they can present proposals for requirements scope change proposals, which may be necessary to reduce project delivery risk.

<table>
<thead>
<tr>
<th>Process: Business Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Establish the business need for the software deliverable. This is the most important process. Not having a fully defined business case for a project will guarantee its failure. This can be seen in the first Fisher-Rosemount Case study.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Objectives:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Establish the business need for the software deliverable. Not having a fully defined business case for a project can jeopardize its success.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Techniques:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Objective (1) can be established by first defining the type(s) of contribution that the project will make. Types of contribution typically fall in to three categories: improvements in sales, cost saving or maintaining market position. Cost savings can either be for the customer, in terms of ownership, or for the company in terms of future development costs or reduced maintenance and support. If the software is to be used internally within the company, then the cost savings are associated with improved productivity. The main thing is to establish the benefits in a quantifiable way to ensure that the reasons for the project are sound i.e. that it's not just a bunch of enhancements that are taken from the products wish-list.</td>
</tr>
</tbody>
</table>
Process Description & Objectives

This sort of release usually carries a minor release revision number against it, such as P2.n. This kind of project can be very expensive, and contribute little to the companies overall business success.

*This should be a section in the Project Management Plan document or a separate document created by the project manager on behalf, and with direct contribution from, the project stakeholders.*

- **Process: Project Objectives**

The objectives of a software project need to be defined, both in terms of how they fit with the company objectives (above) and in terms of the development process improvement.

- **Objectives:**
  1. The project should be planned with regard to delivery on an agreed and achievable date.
  2. The project deliverables should make a significant contribution to the company objectives.
  3. The project must set out its goals with respect to improvements in one or more of its development processes.

- **Techniques**

  1. Objective (1) is achieved by Project Management ensuring that they take a strong position with respect to the control of requirements (scope), taking into account degree of difficulty and availability of the right numbers and skill level of resources. Remember that Marketing's job is to push the development team for as many features as possible. If a development team has infinite time and resources, they would only provide 50% of what Marketing would like. Therefore, be prepared to negotiate hard and fight off the Marketing view that late with loads of...
Process Description & Objectives

<table>
<thead>
<tr>
<th>functionality is somehow a good thing.</th>
</tr>
</thead>
</table>

2. Objective (2) is addressed by the company objectives and business sections above.

3. Objective (3) is achieved by holding project post-mortems following the completion of each software project. From this, a set of process improvements should be factored in to the next project. However, improvements in the process must also take place throughout the project. Never be afraid to change the process if it makes it more efficient or improves the deliverables to the customer.

*This should be a section in the Project Management Plan document.*

- **Process: Project Approval Request Process**

This is the official document, which grants the project the right to expend company funds, in the creation of software and the ultimate release of software deliverables.

- **Objectives:**

  1. To gain approval from all responsible parties for the software project to start expending company funds.

  2. To provide all responsible parties with sufficient project details that they feel comfortable with granting approval to precede.

- **Techniques:**

  1. This technique covers both objective (1) & (2) above. The most economically way to approach this is to initially create a Project Management Plan, with sufficient sections to enable those responsible for approving funds, to attend a review meeting and ultimately give approval for the funds to be employed in the execution of the project. In PRINCE2
Process Description & Objectives

this is known as a Project Approval Request document or PAR. The following indicate which sections need to exist as a minimum in the initial project management plan document. Notice the sections are taken from the list of processes in this table:

- The Business Case (including development and running costs)
- Project Objectives
- Release Strategy
- Resource Strategy
- Risk Management
- Requirements
- Process Capacities
- Process Improvements

- **Process: Requirements Definition**

This is the process whereby customer requirements are captured and balanced against the organisation and project objectives.

- **Process Objectives**

  1. To capture and fully understand the customers real needs.
  2. To balance the customers needs against the organisation objectives and capabilities.
  3. To translate the customer's requirements into technical solutions that will satisfy their real needs and match the project or product goals.
  4. To perform both forward and backward requirements traceability

- **Process Techniques**

  This page contains the following techniques:
**Process Description & Objectives**

- **Objective (1)** can be achieved by holding regular customer focus group meetings. The attendees from the customer side need to be the ‘doers’ i.e. the people who are actually going to use the software product deliverables. From the Software Company the attendees should be the Project Manager, Marketing representative, Team Leaders and selected Technical Architects. This must not be a political meeting with customer’s bring along their favourite list of product issues. The customers must be provided with a strict agenda for the meeting and materials in advance to allow them to prepare. A questionnaire type approach before the meeting is a good technique to use in order to focus the customer’s attendees in the areas of interest. As the requirements / development progresses the group’s members should be fed with requirements definitions, early design, prototypes and early release information / software. Adopting this technique will ensure validation of requirements at every stage in the project.

- **Objective (2)** can be achieved once the customer’s / market needs are established. Once they are established, they need to be balanced against the company’s goals for its current products, future products, and resources and market window. This is a vital step if software disasters are to be avoided such as products that do not meet customer needs, do not contribute to the company’s main direction or miss the market window.

- **Objective (3)** can be achieved using the Quality Function Deployment (QFD) method. The idea behind QFD is the translation of user requirements, expressed in their own, occasionally non-technical terms, into a set of measurable technical specifications. The translation is achieved using a number of matrices and tables, the hierarchical nature of which enables a step-wise refinement of the requirements; move from vague needs to a more detailed understanding. Once the requirements
Process Description & Objectives

are understood the technique will yield a prioritised list in terms of user requirements and their correlation with the quality of the evolving product.

In other words, it gives a chance to review all the proposed enhancements and apply a systematic approach to ranking them in which all stakeholders can participate.

Process Overview: Once the QFD spreadsheet is completed (a template is available from the QFSD repository) it provides a ranking for each project goal, based on the correlation given between each customer requirement request and the project goals. This is balanced by the weighting factor generated by the customer demand column. The customer demand column is driven by a review with customers and marketing.

When the requirement requests have been correlated and the goals have been ranked, the review group then prioritises the requests initially as being in the release or not. Then with the requests that are in the current release, these are given priorities of 1, 2 or 3 and assigned to milestones and subsequently to clusters (development or project groups if it requires a programme of projects) for implementation. In addition each request is assigned an analysis document and software requirements specification in which the request will be formally defined.

Beneficiaries: The first step is to identify the beneficiaries of the project. For example:

- Installed base customers
- New Customers
- Internal Users
Process Description & Objectives

- Marketing / Sales
- Development Team

'What's': These are the customer requirements and will initially be provided in the 'beneficiaries' own words, but must then be refined in to the requirements component parts, and translated in to a precise description. The 'What's are systematically recorded in the QFD spreadsheets 'Features' column. Any important details can be added to each 'What's entry using the cell notes capability provided by EXCEL.

'How's (Goals)': The 'How's are, in this case, the goals of the project. These are generated jointly by the Development Team, Marketing. The following are generic examples of typical goals:

- Move Client Functionality on to the NT platform
- Migrate Database Server Functionality on to NT
- Improve Maintainability
- Project Schedule
- Improve Users Productivity
- Improve ease of installation

Any number of project goals can be set. However, in order to focus the prioritisation it is recommended that the number of goals do not exceed ten.

Correlation:

The next stage is to establish the correlation between the How's & What's i.e. the users requests and the goals of the product as a whole. For example, a user requirement may be a high productivity gain, but could impact the stability of the product if implemented with current technology. Correlation will also contain some 'soft' information i.e. how the change would be
Process Description & Objectives

accepted by the organization as a whole.

The correlation values used are 9/3/1/0 (Strong, Medium, and Weak, No Correlation). These weighting values are taken from the published QFD methodology.

The development team (project manager, team leader, developers) carry out the correlation. This is achieved by holding a series of meetings at which different areas of the 'What's are discussed and weighted against the overall project goals. All actions, open issues, technical clarifications are added to the cell notes associated with each 'What'.

Meeting minutes should also be produced to record the discussions and to prove that the review has been carried out. The results should be published from the final meeting and distributed to senior management and marketing.

Importance To The User (I):

The next step is to review the QFD spreadsheet with Marketing to have them complete the 'Importance to the user', 'Sales Point' and 'Performance Ratio' weighting factors. Each 'What' will be examined with respect to its importance to the user, and weighted accordingly. The importance to the user values is: 1 to 5; (1 being low, 5 being high).

Sales Points (S):

The sales potential of the enhancement will need to be weighted by marketing. The sales values are 1.5, 1.2, 1 (Important, Less Important, assign (1) to all the rest as default).

Performance Ratio (R):

Each 'What' must be compared against the current performance provided by the product and performance improvement provided by the proposed enhancement. This should be expressed as a decimal fraction i.e. 50% improvement = 0.5.
Process Description & Objectives

Calculations:

Once the basic absolute weights have been assigned, the normalised weights need to be calculated. The aim is to come out with a set of prioritised enhancements, together with their correlation to the main goals of the project. The spreadsheet will automatically carry out the calculations. However, there may need to be some adjustments made to the range of rows used by the calculations.

Absolute Weight of each 'What'. \( W = I \times S \times R \)

*The absolute weight column \( W \), is normalised to create the user demand ranking column, which is \( D \).*

Ranking The 'How's

The correlation values in each 'WHAT' row are multiplied by the corresponding value in the User Demand column \( D \). Each 'HOW' column is then summed to get the absolute weight. The absolute weight column is then normalised to rank the project 'How's.'

Enhancement Ranking:

Once the QFD sheet is completed, you will be able to see how each requirement maps against user demand and project goals. The results from all areas will then be transferred into a single spreadsheet to do the overall prioritisation.

The column \( (D) \) gives the basic user demand for a particular requirement. The N.Weight row puts the 'How's, or project goals, into a priority order. From this information you can prioritise the requirements list by looking at the user demand level and how that fits with the project goals. For example: a requirement may have a high customer demand, but does not fit in with the projects goals or fits in with a goal which has been ranked as a low priority.
### Process Description & Objectives

The ranking of the project goals also gives us feedback on the thought processes (or prejudices) we used when assigning the correlation values.

Remember that this final stage will be carried out by the project manager, senior management and marketing.

**Objective (4)** can be achieved by recording the requirements in such away that all interested parties can easily review them, and so that they can be traced. In simple terms, requirement traceability is the process whereby all requirements in a project are traceable through all phases of the project.

There are a number of possible approaches to tracking requirements. The starting point is usually to produce a Software Requirement Specification, or set of specifications, in which all requirements are expressed in unambiguous, non-product specific terms. Each requirement should be given a unique identifier, which is used to track the requirement through all phases of the development. Embedding some intelligence in to the requirement numbering scheme helps with traceability.

A useful way of tracing requirements is to enter them in to a requirements database. This enables them to be tracked at each phase i.e. Design, Code, Test. The requirements database would contain a table for each development phase, against which their design, code and test can be tracked. It is worth bearing in mind that whilst requirements traceability is extremely necessary, you can put to much administrative overhead in to traceability for very little improvement (law of diminishing returns).

The QFD approach to requirements capture and producing formal Software Requirements Specifications, complete with a unique requirements numbering scheme, is a cost effective approach. Whilst the requirements are crosschecked against the design, the absolute must is to have a method of holding each requirement against its equivalent release test case.
Process Description & Objectives

ensures that each requirement can be tested and check off as tested during the final release testing phase. A simple method to achieve this is to write a set of Test Case Procedures (Test Procedure Specifications). Do not embed the requirements numbers against the test cases in these procedures, as this will prevent them from being used for regression testing in a later release. Instead, create a spreadsheet or database, in which each test procedure, set of test cases, associated requirements and test results are recorded. In this way you can be certain that all requirements have been tested.

Process Tools

This page contains hypertext links to the tools templates required to implement the techniques as defined in the process technique section. A template for the QFD spreadsheet can be retrieved from the QFD Requirements Capture & Analysis Spreadsheet. The process capability database can be accessed from the Process Capability Database.

Process Background

The aim of requirement capture is to establish a common understanding between the customer and the software project of the customer's requirements. The customers can be defined as internal or external to the organisation.

Note: Requirements creep is a major contributor to late and unsuccessful projects. Good requirement management is a prerequisite for a successful project. However, some requirement changes will be necessary during a project. These may be due to incorrect initial requirements, technology limitations or business direction changes. In order to cope with requirement changes a project needs to take this possibility in to account. It is recommended that a requirements inflation rate of 1% needs to be built in to the project schedule.
The recommended approach for requirements capture is use Quality Function Deployment (QFD). QFD was established by the Japanese motorcar industry and has been used successfully in manufacturing industries for a number of years. A modified version of QFD is now available for use with software projects. The beauty of the QFD model is that it gives visibility of requirements to all project stakeholders: development team, senior management, marketing and customers. The approach appears to be quite mechanistic, however, the benefits of the technique are listed as follows:

- Provides visibility of requirements and their sources.
- Provides a forum in which all stakeholders are involved in the initial requirement capture, balancing against company and project goals.
- Provides a forum in which to agree the project implementation prioritisation of requirements.
- Provides a vehicle by which requirements prioritisation can be fed back to the customers.
- Provides a process to enable the development teams to understand and document ‘initially’ the requirements, and to consider the impact on the project or product line.
- Provides a baseline of agreed requirements and their relative priorities to feed forward into the project planning stage.

Remember that the sources of the requirements need to be validated to ensure that the request is representative of the majority customer need, and not the pet idea from the development team, marketing or a particularly vocal customer. The QFD analysis, if carried out thoroughly, will enable the project concept (analysis) documents to be written. Remember that each
### Process Description & Objectives

Requirement will be uniquely identified, tracked, tested against and delivered.

- **Process: Executive Test Plan Review**

  This process enables senior management (Development, Marketing and Quality) to ensure that a project's test planning is suitable for the type of product being developed.

- **Objectives:**

  1. To determine that the test planning is suitable for the type of product being developed.

  2. To establish joint responsibility for the planned release testing

- **Techniques:**

  1. The above two objectives can be achieved by holding a meeting with senior management, with materials prepared by the project manager. The materials required will be as follows:

     - An overview of each phase of the software product's release testing. This should include unit, product, system & field trial testing.

     - The project-testing schedule.

     - Description of how the design will be verified by the planned testing.

     - Description of how the product will be validated by customers/users before release.

     - Description of the performance testing that will be applied.

     - The meeting should end with formal approval of the test planning for the product.
Process Description & Objectives

• Process Tools
  
  • A template is provided in the QFSD online repository for the executive test plan structure.

A description of this process is required in the PMP, along with the date of the meeting.

• Process: Executive Release Readiness Review

This process enables senior management (Development, Marketing, Quality, Operations and Technical Support) to ensure that a software deliverable is ready for release.

• Objectives:

  1. To determine if the product is ready to be released.
  2. To ensure that the organization as a whole is ready to support the product release.
  3. To establish joint responsibility for the release of the product.

• Techniques:

The above three objectives can be achieved by holding a release meeting with senior management, with materials prepared by the Project Manager. The materials required for the meeting are as follows:

• Provide a clear understanding of the software product that is to be released.

• Describe the release plan, i.e. full release, partial release, controlled release.

• Describe what impact the product will make on customers if it fails in operation, and on other products, it interfaces with.

• Present the actual design verification by summarizing all test results.
Process Description & Objectives

- Present the actual design validation by summarizing the feedback from the Beta release and any other customer validation results. Include information on how the product compares with similar competitors products.

- Present the current state of the organization with respect to the provision of the following: Manufacture, Documentation, Training, Pricing and Distribution.

- Present all known risks associated with releasing the product.

The meeting should end in a formal approval to release (this may have a set of conditions associated with the release approval).

- **Process Tools**
  
  - A template is provided in the QFSD online repository for the executive release readiness review structure.

  A description of this process is required in the PMP, along with the date of the meeting.

- **Process: Process Improvement**

  The process ensures that continuous improvement, are made to the software development process.

- **Objectives:**

  1. To ensure that one or more development processes are improved during the execution of each project.

  2. To ensure that changes to the existing processes become the norm, rather than the exception.

- **Techniques:**

  1. Objective (1) can be achieved by carrying out a project post-mortem at
Process Description & Objectives

the end of each project. A set of process improvement actions should result, and be applied to subsequent projects.

2. Objective (2) can be achieved by changing the focus of responsibility for process improvement. A good approach is to make sure that process improvement is the responsibility of the Project Manager & Team Leader. The detailed presentation of the process changes is the responsibility of the person who actually has to carry out the process on a day-to-day basis. What this is doing is pushing control down to the people who actually do the job and can make much more informed changes to the processes.

This needs to be a section in the Project Management Plan.

Management Processes

- Process: Project Management Plan

A communication medium by which a development project is managed. Refer to Appendix E for an example of a project management plan which uses the new core framework structure.

- Objectives:

  1. To provide all project stakeholders with information or pointers to information that will enable them to take informed project decisions.
  2. To act as the source of all project planning, execution information and quality management information.

- Techniques:

  1. The Project Management Plan needs to include the same sections as shown in this table, being completed with information specific to a given project(s). However, where possible project information needs to be
Process Description & Objectives

- Process: Project Development Folder
  
  All project artefacts must be collected together in a Project Development Folder. The project development folder contains all project artefacts and must be under configuration control.

- Objectives:
  
  1. To enable all project information to be identified and accessed easily.
  2. To enable a project to be retrieved from archive if necessary.

- Techniques:
  
  1. The above two objectives can be achieved by having an online project development folder containing the following artefacts:
     - Analysis documents
     - Concept documents
     - Source code
     - Inspection records
     - All development meeting minutes
     - Requirements specifications
     - Design documentation
     - Project management plan
Process Description & Objectives

- Project approval document
- Test design / procedure specifications
- Test logs / performance logs / statistics / summaries
- Milestone / release review meeting minutes.
- Product structures
- User documentation plans
- Field trial plans / results
- Defect tracking systems
- etc.

- Process: Project Planning Process

This is one of the most, if not the most, important process within any software project. A project must be planned for success. The following gives guidelines on how a project can be planned. However, in every case, there is no substitute for experience and historical past project information. A word of warning on the subject of scheduling & estimating tools, both types of tools are useful in the planning process, providing you fully understand how they get to the final estimate or dates. Based on that information, the Project Manager will take the information produced from the planning tools in to account in the project planning. Never assume that the tools will give you the schedule you need. Many inexperienced project managers have presented schedules, which show years, when actually the project is completed in months.

- Objectives
Process Description & Objectives

| 1. | Software project activities and commitments are planned and documented |
| 2. | Software estimates are documented for use in planning and tracking of the software project. |
| 3. | Project schedules are established |

- **Techniques**

Project Planning is a subject by itself. The following is an approach that will meet the above objective. However, it should be born in mind that no two projects are the same, and like every other activity in life, it is impacted by a number of external factors. Refer to the project-planning diagram at the end of the techniques section.

1. It is assumed that the requirements definition for the project is completed, and that the QFD approach has been used as described in the requirements definition section other requirements capture techniques can also be used.

2. The results of the QFD analysis for each major project area are extracted from the QFD spreadsheet and recorded in the initial planning spreadsheet (refer to tools heading to obtain a planning spreadsheet template). Each of the agreed requirements is copied from the QFD spreadsheet, along with any related notes. The team should be then introduced to the project development and delivery approach that will be used on the project.

3. Project milestones are established and assigned delivery dates. These milestones are entered as date columns in the planning spreadsheet. In addition, further columns must be added to record early effort.
4. The project manager meets with the development team to establish and record the following in the planning spreadsheet:

- The milestone in which the requirement will be implemented and delivered. A milestone cannot be longer than four months.
- Effort to implement in terms of person-years and actual cost.
- Assignment of resources to implement each requirement.

5. Based on the above information the Project Manager / Team Leader will produce the first draft project schedule. This schedule shows the major tasks, broken down by area (project cluster), milestone and priority within the milestone and any development dependencies. Whilst creating the schedule priorities may be adjusted to take account of resource numbers and skill levels.

6. The project manager uses historic information from past projects to add the following task estimates to the schedule:

- Inspections
- Defect Removal
- Project Administration
- Project Management
- Test Plan Production
- Test Execution
- Development System Support
7. In parallel with the above activities the project team will write the Analysis and Software Requirements Specifications for each of the main development areas. Following the first inspections of the Analysis and Software Requirements Specifications, the project groups will produce a set of task lists (each task must be less than 10 days in duration). These task lists will also contain the first pass estimates. The basic layout of the task lists can be defined in the standard EXCEL template, which can be found in the process tools section. This template is used as the basis for holding all project task estimates and for tracking changes to estimates and the actual effort required to implement the tasks.

8. The Project Manager / Team Leader and Group Leaders then review the first pass task lists and estimates. This review will consider the validity of the estimates, skills available to execute the tasks, priority assigned from QFD analysis and schedule time available. Based on the review a second set of tasks / estimates will be requested from the project team. In addition, a second draft schedule will be produced.

9. In addition to the second set of tasks / estimates, the Group Leaders will be asked to identify the inter project area dependencies. This information is vital from a planning point of view. If the dependencies are not identified in the planning stage, then groups will be waiting for services from other groups, thus giving them dead time. Dead time means things like defect fixing / wish lists, which is a sure indication that the project has not been planned correctly (all dead time should be at the end of the project).

10. A second meeting will take place to review the tasks / estimates / priorities and inter project area dependencies. The results of this meeting will be used to set up the definitive project task tracking system.
Process Description & Objectives

and project schedule. The results of this meeting will also be used as the basis of minor adjustments to the Software Requirement Specifications before they are finally approved.

11. The estimates and schedule are then base lined in the project development folder, for future project planning reference and comparison with actual performance at the end of the project. The schedule and costs are reviewed with senior management before the project moves to the design and code phases.

12. Whilst the above process establishes the planning information for the entire project. The scope, task breakdown and task estimate will be revised before each milestone is started. This will give better estimates and enable effective mid course corrections to be made.

During the above planning process the Project Manager will have to take in to account the available skill levels of the resources available, the future direction goals for the software product and the optimum delivery date.
Process Description & Objectives

• Process Tools

1. A template for the initial project planning spreadsheet template is provided in the QFSD online repository. The template would typically have the following columns:
   - Unique Task ID
   - Short Task Description
   - Long Task Description
   - Design Estimates / Actuals
   - Code Estimates / Actuals
   - Module Test Estimates / Actuals
   - Resource Assignments
   - Priority / Milestone Designators
   - Calculated Fields (Total Estimated / Actual Effort / Effort Remaining / % Progress / Timeline Remaining)

• Process Background

A project must be planned to succeed i.e. the project manager must "stack the deck". There are three simple keys to the successful delivery of a software project:
   - Establish an achievable schedule.
   - Manage requirements scope.
Process Description & Objectives

- Deliver validated requirements.

Establishing an achievable schedule depends to a degree on the more obvious factors such as resource availability, skill levels, accurate estimating, good project control etc. However, by far the biggest danger is trying to meet other stakeholders perceived needs. The pressure on the project manager to agree to a certain scope/time scale, before even the first set of developer estimates have been made is immense. This pressure comes from senior management, marketing and customers. There are only two ways to deal with this pressure and neither of them is pleasant.

1. The first approach is to give a range of dates (best case/worse case) and refuse to give a more concrete schedule until the development team has fully understood the requirements and the planning phase is completed. This approach may not be that successful, therefore if you are forced in to giving dates, you will have to work on prioritised features.

2. The second approach can be applied in the schedule end date had been fixed. Having a set end date forces the project to prioritise the features in to categories, only guaranteeing to deliver on the high priority items. The priority of the items can be found from the QFD analysis spreadsheet. This approach needs to be tempered to ensure that the delivery of the priority one items alone would meet the basic requirements of the business case.

Management of requirements scope is very difficult. Most late requirements are presented to the project manager by marketing types, who claim that the whole product will fail if their feature is not included. The project manager needs to ensure that the requirement has been validated with the customers and is not just the marketing person's
Process Description & Objectives

hobbyhorse. This is where having a strong business case comes in to its own. The additional late requirements can be weighed against the business objectives and will then stand or fall on their merits.

All project requirements must have been first validated against the business needs of the company and the customers. If a product does not satisfy these two need sets, then it can never be classified as a commercial success, even if it is a technical triumph.

• Process: Release Strategy

The release strategy addresses the smooth introduction of the software product to the customers. It therefore has to address all aspects of the release e.g. software manufacture, documentation etc.

• Process Objectives

1. To release the software product to the customers with minimum disruption.

2. Ensure that the organization is ready to release the software product.

• Process Techniques

1. Objective (1) can be achieved by making sure that the impact on the customer of the new software product is fully understood. The items to be considered are: what will the customer be required to purchase or change to accommodate the new product? What will the migration issues be? What support in terms of documentation or training will the customer need to install and make the most effective use of the product?

2. Objective (2) can be achieved by setting up an Interdepartmental team
Process Description & Objectives

early in the project. This team consists of representatives from all departments involved in the software product development e.g. manufacturing, marketing, sales, education, documentation etc. This team must meet on a regular basis to plan and coordinate the introduction of the product.

- **Process: Resource Strategy**

Availability of suitably qualified resources is always a major issue when planning projects, especially if there are a number of projects competing for the same resources. Obviously, the types of resource required varies depending on the type of project and the extent to which the management of development is outsourced to a third party or indeed parties. Management of third party resources is difficult and requires close project management co-operation between the project owners and the third party. This is even more challenging when the third party is offshore. However, generally the progressive ramp up on projects and gradual winding down shows a profile that is in line with a well-planned project. This preferred resource profile is shown in the second case study.

- **Objectives**

1. Apply resources to a software project in such a way that the resources are fully occupied and are not overloading existing resources
2. Apply the correct level of skills to the project
3. Apply a central core of experienced staff to the project
4. Manage the resources such that they are motivated and simply given one specific area of work. The clusterisation approach fosters this approach, by making the cluster responsible for all changes in the
Process Description & Objectives

system required to deliver the assigned vertical slice.

This contains the staff plan, team structures and the strategy for resource loading.

- Process: Requirement Capture

Once the requirements have been defined using the QFD process, it is recommended that the requirements are either expressed as very concise text or using the universal modelling language (UML).

- Process: Development Process

The development approach needs to ensure that the processes used to develop the software deliverables, are the most efficient for that specific type of software product. Whatever detailed processes are used, the following general principles will prove effective in obtaining predicable schedules. These principles have been used very successfully by the researcher on six large software projects (two of which were new product releases). However, if a methodology such as the unified process is used, then it can be integrated into the framework in preference. An example of this integration is shown in Chapter 10.

- Objectives

1. To simplify a software system in order that it can be developed successfully.

2. To ensure that resources are managed efficiently.

3. To ensure that the development is easily controlled.

4. To ensure that historical estimates are captured and easily reused.

5. To avoid the one-man-on base situation.

- Techniques
Process Description & Objectives

1. Objective (1) can be achieved by using a very simple engineering approach. If a system is complex, then apply a functional decomposition approach. In other words, divide the problem up into a series of smaller problems, which can be more easily solved. Each of these smaller units will need to be integrated into an overall system and tested (this will be described under the implementation processes section). There are a number of ways the decomposition can be done. However, broadly speaking, there are two basic approaches. The first would be applied to the initial release of a software product. In the case of a new product, the tasks are to develop the fundamental architecture. The decomposition would be done on architectural lines i.e. database design, user interface, administration functions etc. In the case of an enhancement to an existing product, then a vertical slice containing a specific new feature (or group of features) is assigned to a development group. All groups work concurrently, and can be in different phases i.e. Requirements Translation, Design, Code or Module Test.

2. Objective (2) is achieved by assigning one (or more) of the functionally decomposed units to a specific development group. An individual group should not consist of more than four Developers and a Team Leader. Experience has shown that when a development group gets to above five Developers, the management overheads, communications and work directive difficulties start to increase exponentially. Groups over this size can still be effective, but require Team Leaders with good technical, project execution and personal management skills (these types of people are not abundant). Each group is responsible for the Requirements Translation, Estimates, Design, Code, Module Test and Quality of their group's deliverables.

3. Objective (3) is achieved by having each group produce its own
### Process Description & Objectives

estimates, task breakdowns and record actual effort. The information is held in electronic form and updated by the Team Leaders on a weekly basis. This enables the Project Manager to be sure that each group has planned their work by doing a task breakdown and estimates. In addition, progress information can be collected easily. The task information contains a description of the task, the resource assigned, its estimated effort, actual effort, progress, milestone completed against and status.

4. Objective (4) is achieved as each groups task tracking details are archived at the end of each project, and can be used to validate subsequent project task estimates.

5. Objective (5) is to avoid having resources, which are specialists in only one area. This is an easy situation to get into, but difficult to resolve. Using the vertical slice approach, where a group is given responsibility for a ‘slice’ of functionality. This requires each group to have a broad range of development skills. To achieve this capability requires a training investment, but the benefits are well worth the effort.

---

### Process: Risk Management Process

This section details the project constraints and risk management strategy for the currently known project risks. In addition, alternatives considered need to be described. Risk management is the process of measuring, or assessing risk and then developing strategies to manage the risk. In ideal risk management, a prioritisation process is followed whereby the risks with the greatest loss and the greatest probability of occurring are handled first, and risks with lower probability of occurrence and lower loss are handled later. In practice the process can be very difficult, and balancing between risks with a high probability of occurrence but lower loss vs. a risk with high loss but lower probability of occurrence can often be
Process Description & Objectives

Risk management also faces a difficulty in allocating resources properly. This is the idea of opportunity cost. Resources spent on risk management could be instead spent on more profitable activities. Again, ideal risk management spends the least amount of resources in the process while reducing the effects of risks as much as possible.

- Objectives
  1. The objective is to manage risks within an acceptable cost and timescale.

- Techniques
  Once risks have been identified and assessed, all techniques to manage the risk fall into one or more of these four major categories:
  1. Avoidance
  2. Reduction
  3. Retention
  4. Transfer

  Ideal use of these strategies may not be possible. Some of them may involve trade offs that are not acceptable to the organization or person making the risk management decisions.

Risk avoidance

Includes not performing an activity that could carry risk. An example would be not buying a property or business in order to not take on the liability that comes with it. Another would be not flying in order to not take the risk that the plane could be hijacked. Avoidance may seem the answer to all risks, but avoiding risks also means missing the potential gain that accepting (retaining) the risk may have allowed.
Process Description & Objectives

Not entering a business to avoid the risk of loss also avoids the possibility of earning the profits.

**Risk reduction:**

Involves methods that reduce the severity of the loss. Examples include sprinklers designed to put out a fire to reduce the risk of loss by fire. This method may cause a greater loss by water damage and therefore may not be suitable. The Halon fire suppression systems may mitigate that risk, but the cost may be prohibitive as a strategy.

**Risk retention:**

Involves accepting the loss when it occurs. True self insurance falls in this category. All risks that are not avoided or transferred are retained by default.

**Risk transfer:**

Means causing another party to accept the risk, typically by contract. Insurance is one type of risk transfer. Other times it may involve contract language that transfers a risk to another party without the payment of an insurance premium. Liability among construction or other contractors is very often transferred this way.

Once the risks are put into the various categories, a risk management plan is created for the project. This risk plan is actively managed by the project manager and reviewed / updated at regular intervals during the project.

The risk management plan will contain the following:

- Unique identifiers for referring to the same risk in company or project documents (*identification*)
- Describing the risk and how it could become a liability (*description*)
- Assessing the consequences of that (*effect*)
Process Description & Objectives

- Describing what precautions could be taken to prevent it (precaution)
- Drawing up contingency plans or procedures for handling it (contingency)
- Categorising the risk as new, ongoing or closed (risk status)
- Estimating the probability of the risk becoming a liability (Risk escalation probability, $P$)
- Estimating the consequences in terms of time for the project (Schedule impact, $S$)

In addition, every probable risk must have a pre-formulated plan documented to deal with it to deal with its possible consequences (to ensure contingency if the risk becomes a liability). The risk management plan will have a spreadsheet associated with it, which enables the risks to be sorted by cost or schedule impact, using the following simple ratio calculations.

From the information above and the average cost per employee over time, or Cost Accrual Ratio, a project manager can estimate

- The cost associated with the risk if it arises, estimated by multiplying employee costs per unit time by the estimated time lost (cost impact, $C$ where $C = CAR \times S$)
- The probable increase in time associated with a risk (schedule variance due to risk, $Rs$ where $Rs = P \times S$):
- Sorting on this value puts the highest risks to the schedule first. This is intended to cause the greatest risks to the project to be attempted first so that risk is minimised as quickly as possible.
- This can be slightly misleading, as schedule variances with a
### Process Description & Objectives

<table>
<thead>
<tr>
<th>large P and small S and visa-versa are not equivalent.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>• The probable increase in cost associated with a risk (cost variance due to risk, ( R_c ) where ( R_c = P \times C = P \times CAR \times S = P \times S \times CAR ))</strong></td>
</tr>
<tr>
<td><strong>• Sorting on this value puts the highest risks to the budget first.</strong></td>
</tr>
</tbody>
</table>

Risk in a project or process can be due to special causes of deviation or common causes of deviation and requires appropriate treatment.

**• Process Tools:**

There are a number of risk management techniques that can be plugged in to the framework such as:

- Consultative, Objective and Bi-Functional Risk Analysis COBRA' (2005).

**• Process: Monitoring & Control**

Details on task breakdown / effort / progress tracking / progress reporting.

**• Process: Project Review**

This process is to ensure that the project is reviewed both internally and externally, throughout its development.

**• Objectives:**

1. To ensure that the project is executed as planned by frequent reviews
2. To be in a position to take early corrective action to bring the project back on track
3. To provide an opportunity for continuous monitoring of the quality of both the evolving software product and the processes used in
### Process Description & Objectives

#### its creation

- **Techniques:**

  - **Project Approval Request review (PAR)**

    The PAR documents are reviewed across all departments impacted by the project and up to Vice President level. The review materials are located in the QFSD online repository.

  - **Project management plan review**

    The project management plan is reviewed across all departments impacted by the project; senior technology management and the proposed inter departmental team members. The review materials are again located in the QFSD online repository.

  - **End of milestone project reviews**

    At the end of each project development / test milestone, a review is held with the project manager cluster leaders.

    The reviews focus on the following:

    - Establish if the main milestone objectives have been achieved
    - Review that all scheduled tasks have been completed, by looking at the tracking spreadsheet. Tasks not completed are moved to subsequent milestones (If applicable)
    - The inspection utility is reviewed to ensure that all inspections for the milestone have been completed, and all associated actions completed
    - A summary of the effort / costs employed in achieving the milestone are reviewed, and the project cost tracking spreadsheet is updated
Process Description & Objectives

- A summary of the estimate changes since the milestone started is recorded to help with the next milestone planning.

- Actions are set for the planning of the subsequent milestone.

At the end of each milestone, a formal sign off sheet is generated and recorded. The milestone review meeting materials are made available prior to the review in the QFSD online repository.

- Executive test plan review:

This is described in an earlier section. The meeting materials are also recorded in the QFSD online repository.

- Executive release readiness review:

This is described in an earlier section. The meeting materials are recorded in the QFSD online repository.

- Monthly V.P / director level project reviews:

These reviews take place on a monthly basis and serve to review the project against other projects, and the overall organizational direction.

- Inter departmental team project reviews:

Regular meetings are held with the inter-departmental team (IDT) to review major project changes and review modifications to the project management plan document. The IDT team develop the product support, manufacturing, Education, ordering and documentation plans. All IDT groups consist of representatives from each of the major functions outside the IT group.

These can include:

- Operations
- Manufacturing / NA / EMA
Process Description & Objectives

- Technical Documentation
- Product Marketing
- Drafting
- Purchasing
- Technology

The IDT meeting is chaired by the project manager. As the project approaches release, these IDT meetings will become more frequent in order to make sure all departments are prepared and trained for the release of the software deliverable.

- **Process: Test Management**

  The principle is that all test cases must be created as early as possible in the project. Once the functional / non-functional requirements are defined the corresponding test cases should be created. Each test case must be mapped to its corresponding requirement or requirements. This will ensure that all requirements are verified during the testing phase. All test documents must be inspected with the same rigour as the project functional and non-functional requirements documents.

- **Objectives:**
  1. To verify all requirements within the software system
  2. To ensure that all test materials are reviewed and tracked

- **Techniques:**

  The first technique has been discussed above in terms of making sure that each test case is mapped to its corresponding function or non-functional requirement. This mapping is best done using either a database
Process Description & Objectives

or spreadsheet remembering to archive the spreadsheet or database in to the QFSD online repository at the end of the project. Test materials are reviewed and tracked in the QFSD online repository.

In order to determine when the software is stable in terms of defect discovery rate, a preset metric of defects levels per test hour is established at the start of the project and monitored during each phase of testing.

Testing phases vary based on the nature of the project, but generally consist of unit, product, system, performance and pilot. Each test phase must have a detailed review and phase acceptance criteria defined in the top-level test design plan for the project.

Typical test artefacts for a project are:

1. Top level test design specification
2. Individual test procedure specification
3. Test logs for each test procedure specification
4. Test summary and performance benchmark results
5. Defect rate metric tables and corresponding graphs
6. Automated test scripts and test logs (if applicable)

Defect rate graphs are used to measure the stability of a software product that is under test in order to determine when it is ready for release (or transition to the next phase). The defect rate metrics are defined in the test design specification prior to the start of the first test phase.

• Process: Problem Resolution

Problem resolution is straightforward, but does require the ground rules to be set for the project. Problems are defined as those occurrences that impact the project, but are outside of the projects direct control.
Process Description & Objectives

- Objectives:
  1. To manage or minimise the impact of problems that are outside of the project's direct control

- Techniques:

  Unlike risk management, which identifies probable events that may impact the project and puts in place plans to handle these situations, this process handles events that were not identified and that could be major showstoppers for the project. In reality, the types of problems that can affect a project can be identified from experience, at least in general types.

  From the researchers' experience these are:

  - Investment levels are cut back due to a poor company performance. In which case the project manager must plan the project such that the product can be delivered at the end of each development phase, just in case there are no subsequent phases.

  - Major failure occurs in the development equipment or the site itself. The project manager must ensure that the correct level of disaster recovery is in place to enable the project to continue in the event that a disaster occurs. Never assume that disaster recovery plans are in place, or are effective. The project manager must always ensure that the project can recover in the minimum time possible.

  - A supplier of a major software component ceases trading or removes support. This requires the project manager to be in constant contact with any components suppliers and to ensure that a Dunn and Bradstreet financial review is carried out on the major suppliers to the project during the planning stage. This may seem over cautious, but given that the majority of small to medium sized software companies...
### Process Description & Objectives

are not in profit and are funded by venture capital, it is highly likely that this type of problem will occur at some point.

---

### Support Processes

- **Process: Configuration Management**

  This contains details on how each project artefact is traced and maintained. Depends on the configuration management tool selected by the specific project.

- **Process: Document Management**

  Based on the QFSD online repository.

- **Process: Quality Assurance**

  This contains the review structure for ensuring the project stays on track and meeting its goals. This is defined for the specific project within the Project Management Plan document.

- **Process: Verification / Inspections**

  Overview of processes, which will ensure that all software functions are correct.

  The preferred approach is to use inspections rather than reviews. Inspections based on peer groups with an efficient QFSD online repository for handling the review process and an associated issue is preferred.

  In QFSD there is a preferred process and associated tools, which are described in Chapter 6.

- **Process: Validation**

  Overview of processes which will ensure that we are delivering the functions
# Process Description & Objectives

the customers want.

Again this is specific to a project and is described in the relevant project management plan.

- **Process: Third Party Software & Re-Use**

Details the third party products used in development & delivered to the customers.

This process should be part of a company’s IT strategy. The decision to use third party technical or business applications is usually defined in the IT strategy. In the same way the goal of reusing technical services, such as standard database access routines, is usually recognised by any development team that develops software in-house. However, achieving reuse at the business services level is part of the IT strategies alignment with the business strategy e.g. reusing a replenishment function between different business areas. This area will be the subject of future research in terms of adding a further layer to the QFSD model in order to achieve enterprise architecture.

- **Process: Defect Management**

Details how issues are found, logged and their resolution priority set. Again, this is a standard process and approach that should be referenced in the specific project management plan.

- **Process: Process Improvements**

This is based on two further extensions to the QFSD framework. That of process performance measurement and process capability. As these form part of the extended QFSD model they have been included in Chapters 6 and 8.

- **Process: Change Control**
Process Description & Objectives

Problem Reporting System / Problem Resolution and Change Control.

Again a standard change control procedure should be used and referenced within the project management plan.

Implementation Processes:

- Process: Approach

Approach to be used for the implementation is dependant on the development methodology that is adopted. Currently, the preferred methodology is to use the Unified Process (UP), which covers the majority of the processes in this section. If the Rational version of UP is used (RUP) then the guidelines and templates are easily integrated into the QFSD online repository. This was done for the Celesio UK wholesale business unit AAH Pharmaceuticals.

- Process: Requirements

Requirements capture phase process model (based on UP).

- Process: Software Design

Design phase process model (based on UP).

- Process: Software Construction & Unit Test

Code & unit test phase process model (based on UP).

- Process: Defect Repair Management

Based on UP or other standard approach.

- Process: System Integration

Release testing phase process model (based on UP).

- Process: Software Test
**Process Description & Objectives**

Usually this is tailored, based on a specific product type and needs a separate test design specification.

- **Process: Organisational Readiness**
  Organizational release phase process model.

- **Process: System & Software Maintenance**
  Organizational release phase process model.
# Appendix D Project Management Questionnaire Results

The following questionnaire is described in Section 4.1.

<table>
<thead>
<tr>
<th>Project Management Method, Tool or Technique</th>
<th>Abs Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project Management Methods/Methodologies</strong></td>
<td></td>
</tr>
<tr>
<td>Projects in controlled environment (PRINCE)</td>
<td>23</td>
</tr>
<tr>
<td>Projects in controlled environments 2 (PRINCE2)</td>
<td>14</td>
</tr>
<tr>
<td>Structured systems analysis and design methodology (SSADM)</td>
<td>17</td>
</tr>
<tr>
<td>The European risk management methodology (RISKMAN)</td>
<td>1</td>
</tr>
<tr>
<td>The RIBA plan of work</td>
<td>2</td>
</tr>
<tr>
<td>Other project management methods/methodologies</td>
<td>16</td>
</tr>
<tr>
<td>In-house project management methods</td>
<td>128</td>
</tr>
<tr>
<td>In-house similar to PRINCE</td>
<td>5</td>
</tr>
<tr>
<td><strong>Project Management Tools</strong></td>
<td></td>
</tr>
<tr>
<td>Critical Path Method (CPM)</td>
<td>70</td>
</tr>
<tr>
<td>Work breakdown structure (WBS)</td>
<td>75</td>
</tr>
<tr>
<td>Cash flow analysis (CFA)</td>
<td>43</td>
</tr>
<tr>
<td>Gantt bar charts</td>
<td>152</td>
</tr>
<tr>
<td>Graphical evaluation and review technique (GERT)</td>
<td>4</td>
</tr>
<tr>
<td>Programme evaluation and review technique (PERT)</td>
<td>24</td>
</tr>
<tr>
<td>Strengths, weaknesses, opportunities and threats (SWOT)</td>
<td>41</td>
</tr>
<tr>
<td>Other project management tools</td>
<td>21</td>
</tr>
<tr>
<td>Project management software</td>
<td>182</td>
</tr>
<tr>
<td>In-house project management tools</td>
<td>5</td>
</tr>
<tr>
<td><strong>Decision Making Techniques</strong></td>
<td></td>
</tr>
<tr>
<td>Cost benefit analysis (CBA)</td>
<td>88</td>
</tr>
<tr>
<td>Decision analysis (DA)</td>
<td>9</td>
</tr>
<tr>
<td>Sensitivity analysis (SA)</td>
<td>19</td>
</tr>
<tr>
<td>Expressed preferences</td>
<td>23</td>
</tr>
<tr>
<td>Project Management Method, Tool or Technique</td>
<td>Abs Result</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>Implied preferences</td>
<td>11</td>
</tr>
<tr>
<td>Revealed preferences</td>
<td>11</td>
</tr>
<tr>
<td>Other decision making techniques</td>
<td>9</td>
</tr>
<tr>
<td>In-house decision making techniques</td>
<td>2</td>
</tr>
<tr>
<td><strong>Risk Assessment Tools</strong></td>
<td></td>
</tr>
<tr>
<td>Lifecycle cost analysis (LCCA)</td>
<td>25</td>
</tr>
<tr>
<td>Event tree analysis (ETA)</td>
<td>8</td>
</tr>
<tr>
<td>Fault tree analysis (FTA)</td>
<td>6</td>
</tr>
<tr>
<td>Probability analysis (PA)</td>
<td>34</td>
</tr>
<tr>
<td>Reliability analysis</td>
<td>13</td>
</tr>
<tr>
<td>Uncertainty analysis</td>
<td>3</td>
</tr>
<tr>
<td>Failure mode and effect analysis (FMEA)</td>
<td>10</td>
</tr>
<tr>
<td>Hazard analysis (HA)</td>
<td>9</td>
</tr>
<tr>
<td>Hazard and operability studies (HAZOP)</td>
<td>9</td>
</tr>
<tr>
<td>Operation and maintenance risk analysis (OMRA)</td>
<td>4</td>
</tr>
<tr>
<td>Preliminary hazard analysis (PHA)</td>
<td>5</td>
</tr>
<tr>
<td>Other risk assessment tools</td>
<td>7</td>
</tr>
<tr>
<td>In-house risk assessment tools</td>
<td>14</td>
</tr>
<tr>
<td><strong>Computer models/databases/indexes</strong></td>
<td></td>
</tr>
<tr>
<td>CRUNCH</td>
<td>1</td>
</tr>
<tr>
<td>Lessons learnt files (LLF)</td>
<td>23</td>
</tr>
<tr>
<td>Expert Systems</td>
<td>4</td>
</tr>
<tr>
<td>In-house computer models/databases/indexes</td>
<td>12</td>
</tr>
<tr>
<td><strong>Computer simulations</strong></td>
<td></td>
</tr>
<tr>
<td>Hertz</td>
<td>1</td>
</tr>
<tr>
<td>Monte Carlo</td>
<td>10</td>
</tr>
<tr>
<td><strong>Other techniques</strong></td>
<td></td>
</tr>
<tr>
<td>Other techniques</td>
<td>17</td>
</tr>
</tbody>
</table>
18 Appendix E Project management plan template

One of the major templates associated with the QFSD framework in the project management plan. The structure of the project management plan is shown below. However, it should be noted that, in the actual online template, each section is supported by a guidance notes to ensure that the template is completed consistently. Given that the template is large and over a third of it is really a description of the project management lifecycle model then the template and guidance information was one of the first documents to be moved in to the QFSD online repository.

1. Introduction
   1.1. Purpose
   1.2. Scope
   1.3. Audience
   1.4. References
   1.5. Development Stages

2. Integral Stages

2.1. Project Management

2.1.1. Project Management

2.1.2. PRINCE2 Methodology

2.1.2.1. Benefits of using PRINCE

2.1.3. PRINCE Project Organisation

2.1.3.1. Programme Management Board

2.1.3.2. The Project Board

2.1.3.2.1. Project Executive

2.1.3.2.2. Senior User
2.1.3.2.3. Senior Supplier

2.1.3.3. The Project Team

2.1.3.3.1. Project Manager

2.1.3.3.2. Application Development Manager

2.1.3.3.3. Production Manager

2.1.3.3.4. Help Desk / Support Manager

2.1.3.3.5. Business Analyst

2.1.3.3.6. Application Development Team

2.1.3.4. Project Assurance

2.1.3.5. Project Support

2.2. Project Development Folder

2.3. Document Templates

2.4. Configuration Management

2.5. Requirements Management

2.6. Project Quality Assurance

2.6.1. Inspections

2.6.2. Quality Documentation Structure

2.7. Corrective Action

2.8. Verification and Validation

2.9. Document Management

2.9.1 Purpose

2.9.2 Definition of Retained Material

2.9.3 Configuration Management System Filing System

2.9.4 Naming Conventions
2.9.5 Version Control

2.9.5.1 Version Numbers
2.9.5.2 Cosmetic Changes
2.9.5.3 Revue Changes
2.9.5.4 Fundamental Changes

2.9.6 Document Versioning Techniques

2.9.7 Document Lifecycle
2.9.7.1 Creation
2.9.7.2 Configuration
2.9.7.3 Change
2.9.7.4 Retention
2.9.7.5 Destruction

3. Project Stages

3.1. Stage 0 - Start Up
3.2. Stage A - Initiation and Planning
3.3. Stage B - Requirements Capture
3.4. Stage C - Functional Spec / High Level Design
3.5. Stage D - Investment Appraisal
3.7. Stage F - Test
3.8. Stage G - Implementation
3.9. Stage H - Closedown
3.10. Stage X - All Stages

4. Stage deliverable & Activity detailed descriptions
4.1. Stage Sign Off VS Required Materials for each Stage

5. Activity Codes

6. Document history
19 Appendix F Lifecycle phase mapping

Analysis to map the identified lifecycle phases and artefact deliverables against the 'core' QFSD framework processes.

Table F1 Lifecycle phases and artefact mapping

<table>
<thead>
<tr>
<th>Lifecycle Phase</th>
<th>QFSD Process Area</th>
<th>QFSD Processes</th>
<th>Lifecycle Deliverables</th>
</tr>
</thead>
<tbody>
<tr>
<td>(O) Start-up</td>
<td>Organisational</td>
<td>Business case</td>
<td>Business case template</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Project objectives</td>
<td>Included in business case</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Project approval request (PAR)</td>
<td>Project approval request template</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Requirements definition</td>
<td>Requirements specification, process modelling templates &amp; tools</td>
</tr>
<tr>
<td>(A) Initiation &amp; planning</td>
<td>Organisational</td>
<td>Process improvements</td>
<td>Process improvement procedure &amp; tool</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Executive release readiness</td>
<td>Not required</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Executive test plan review</td>
<td>Not required</td>
</tr>
<tr>
<td>(A) Initiation &amp; planning</td>
<td>Management</td>
<td>Project management plan</td>
<td>Project management plan</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Project planning</td>
<td>Schedules (Gantt, WBA...)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Release strategy</td>
<td>Release planning</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Resource strategy</td>
<td>Resource estimate</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Risk management</td>
<td>Risk management plan</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Monitoring &amp; control</td>
<td>Project planning tool</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Project review</td>
<td>Progress reports</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Problem resolution</td>
<td>Not required</td>
</tr>
<tr>
<td>(B) Requirements</td>
<td>Management</td>
<td>Requirements capture</td>
<td>Requirements specification</td>
</tr>
<tr>
<td>(C)</td>
<td>Functional specification &amp; high level design</td>
<td>Implementation</td>
<td>Requirements definition</td>
</tr>
<tr>
<td>-----</td>
<td>---------------------------------------------</td>
<td>----------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software design</td>
<td>System integration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Development process</td>
<td>(clusterisation)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Development process</td>
<td>templates</td>
</tr>
<tr>
<td>(D)</td>
<td>Investment appraisal</td>
<td>Management</td>
<td>Investment appraisal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(New process for inclusion in the QFSD framework)</td>
<td>template</td>
</tr>
<tr>
<td>(E)</td>
<td>Detailed design, code &amp; construction</td>
<td>Implementation</td>
<td>Approach definition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software construction</td>
<td>procedure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Detailed design</td>
<td>As above</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Defect repair strategy</td>
<td>Detailed design templates</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Defect tracking tool</td>
<td></td>
</tr>
<tr>
<td>(F)</td>
<td>Test</td>
<td>Management</td>
<td>Test management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Release testing</td>
<td>&amp; test results templates</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Release testing</td>
<td></td>
</tr>
<tr>
<td>(G)</td>
<td>Implementation</td>
<td>Implementation</td>
<td>Organisational readiness</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software maintenance</td>
<td>Software pilot templates</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Infrastructure deployment</td>
<td>Readiness check list</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(New process for inclusion in the QFSD framework)</td>
<td>Hardware &amp; network</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Maintenance plan</td>
<td>deployment templates</td>
</tr>
<tr>
<td>(X)</td>
<td>Supporting processes</td>
<td>Supporting processes</td>
<td>Configuration management</td>
</tr>
<tr>
<td>Document management</td>
<td>QFSD online repository</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>----------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quality assurance</td>
<td>Quality guidelines as provided by QFSD online</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verification</td>
<td>Inspection procedure as provided by QFSD guidelines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validation</td>
<td>Testing approach as provided by the QFSD guidelines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Third party software reuse</td>
<td>As provide by the QFSD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defect management</td>
<td>As provide by the QFSD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process improvement</td>
<td>As provide by the QFSD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change control</td>
<td>As provide by the QFSD</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
20 Appendix G – Case Studies summary and limitations
This appendix provides an overview of the case studies carried out during this research and indicates the limitations encountered whilst using this method.

Table G1 provides a summary of the case studies carried out during this research.

Table G1: Case Study Summary

<table>
<thead>
<tr>
<th>Case Study</th>
<th>Method</th>
<th>Framework Status</th>
<th>Population</th>
<th>Peer Reviews/ Interviews</th>
<th>Primary Data Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case Study One.</td>
<td>Direct Observation (Single)</td>
<td>Conceptual</td>
<td>Project Mgr 1</td>
<td>PR = 1</td>
<td>Project charter.</td>
</tr>
<tr>
<td>Fisher. Rosemount</td>
<td></td>
<td></td>
<td>Team Lead 1</td>
<td>PR = 1</td>
<td>High &amp; detail level specs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mark’g Mgr 1</td>
<td>PR = 1</td>
<td>Test design specs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Developer 20</td>
<td>PR = 5</td>
<td>Test result logs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Testers = 15</td>
<td>PR = 1</td>
<td>Project post-mortem</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Senior Mgt 2</td>
<td>INV = 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Users 5</td>
<td>INV = 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>QA Mgr 1</td>
<td>PR = 1</td>
<td></td>
</tr>
<tr>
<td>Case Study Two.</td>
<td>Direct Observation (Single)</td>
<td>Conceptual</td>
<td>Project Mgr 1</td>
<td>PR = 1</td>
<td>Project charter.</td>
</tr>
<tr>
<td>GEC Power</td>
<td></td>
<td></td>
<td>Team Lead 1</td>
<td>PR = 4</td>
<td>High &amp; detail level specs.</td>
</tr>
<tr>
<td>Instrumentation &amp;</td>
<td></td>
<td></td>
<td>Developer 8</td>
<td>PR = 3</td>
<td>Test design specs.</td>
</tr>
<tr>
<td>Control.</td>
<td></td>
<td></td>
<td>Testers 5</td>
<td>INV = 4</td>
<td>Test result logs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Senior Mgt 3</td>
<td>PR = 1</td>
<td>Project post-mortem</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Users 2</td>
<td>INV = 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>QA Mgr 1</td>
<td>INV = 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PR = 1</td>
<td>PR = 1</td>
<td></td>
</tr>
<tr>
<td>Case Study Three.</td>
<td>Participant Observation (Multiple)</td>
<td>Core QFSD</td>
<td>Project Mgr 1</td>
<td>PR = 1</td>
<td>QFSD repository: templates code and</td>
</tr>
<tr>
<td>Fisher-Rosemount</td>
<td></td>
<td></td>
<td>Team Lead 3</td>
<td>PR = 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Developer 12</td>
<td>PR = 4</td>
<td></td>
</tr>
</tbody>
</table>
20.1 Limitations encountered

Whilst improvements in software schedule, budget and quality were generally observed from the results of Case Studies 2, 3 & 4 and those referenced in Tables: 4.1, 8.1, 8.3, 9.3 & 9.11, there were a number of contributory factors to both the success and limitations of the Case Study approach used.

20.1.1 Issues in the design

There are inherent drawbacks in qualitative Case Studies, such as potential subjective interpretation and difficulties in collecting and pattern recognition across multiple Case Study outcomes. Miles and Huberman (1984) provide guidelines on how to overcome a number of these problems.

Two of the practical problems encountered in the application of the Case Study approach were coordination of multiple Case Study projects and the capture and analysis of data volumes generated.
20.1.2 Coordination and interpretation of data

A very similar set of observations were made in each of the Case Studies, giving opportunities for pattern recognition and, therefore, tight control of Case Study execution was applied. To the extent that the Case Studies were executed as an overall program with tight controls on how the observations and analysis were carried out and documented.

Case Study results are also open to interpretation therefore it is preferable to have two researchers in order to challenge the emerging results. However, it could be argued that having a single mind would be better in terms of overall pattern recognition.

For the first part tight coordination was further achieved by selecting Case Study projects that did not run sequentially. However, for practical reasons of project schedules within the companies this was not always possible.

For the second part Case Study results were always peer reviewed by key senior participants in the Case Study projects. This approach improved the interpretation of the results and also generated further support for the framework as the participants felt more involved in the overall research.

20.1.3 Units of observation

Selection of observation units was based on two criteria:

- Observation would enable the effectiveness of the software process to be evaluated.

- That the source data would be available from basic project management and development artefacts.

The data sources identified were as follows:

- Project management plan
- Project Gantt charts
- Work breakdown charts
- Resource loading charts
- Master requirements list
• Test management monitoring and results
• Defect management monitoring and results

For Case Studies 1 & 3 these sources supported the observation units required. However, in order to have a more rigorous and systematic observation method, an approach based on measuring each framework processes performance and capability was adopted using the tools provided within the framework. Case Study 4 (refer to Chapter 7) provides details on this revised approach, which was applied to all remaining case studies.

20.1.4 Rational for choice of Case Studies

Yin (1984) advocates selecting each additional case in a research programme to address a very specific aspect of theory not addressed in the previous cases. In this research the approach is similar where new processes are being added, but the majority of cases, provide the verification of the framework process by its repeated application in different types of development projects.

The decision criteria upon which projects were adopted as Case Studies were as follows:

• Willingness of the project teams to engage.
• Extent to which the framework could be used in the project.
• Diversity of project type (e.g. new development, enhancement, new technology etc).
• Stable political environment.
• Sufficient and suitably qualified project staff.

20.1.5 Quality of Case Study results

By carefully selecting which projects would be used for the Case Studies, a number of external elements that would artificially skew the results were eliminated. Whilst the Case Study process itself was successful, projects were not all on-time and budget. Rather the repeated application of the framework resulted in a continuous overall improvement in the key software development processes.
and hence improvements in overall accuracy in terms of schedule, budget and user acceptance.

Case Study 4 highlighted that the performance and capability measurement approach needed to be applied to each milestone in the project and not left until the end of the project. Application at the end proved to be suboptimal as issues associated with applying the framework, or in a number of cases, the lack of the right skills in the project, could not be corrected and hence maximum benefits from the framework were not being achieved. Application of the measurements at the end of each project milestone allowed any issues to be corrected early in the project, which gave much improved results from the framework and also gave the project managers more control as a by product.

This continuous monitoring by milestone was adapted into the framework and accounts for a high success rate in the subsequent Case Studies.
### 21 Appendix H – QFSD to ISO 9001 & CMM mapping

#### Table H1: Mapping of QFSD Processes against ISO 9001 and CMM

<table>
<thead>
<tr>
<th>QFSD Processes</th>
<th>ISO 9001 Processes</th>
<th>CMM Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organisational: Company objectives</td>
<td>4.1: Management responsibility</td>
<td>Commitment to perform, Software project planning, Software project tracking and oversight</td>
</tr>
<tr>
<td>Organisational: Business case</td>
<td></td>
<td>Software quality assurance Ability to perform</td>
</tr>
<tr>
<td>Organisational: Project objectives</td>
<td></td>
<td>Verifying implementation Software quality management</td>
</tr>
<tr>
<td>Organisational: Project approval request</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organisational: Exec test plan review</td>
<td>4.2: Quality system</td>
<td>Verifying implementation Software project planning Software quality assurance</td>
</tr>
<tr>
<td>Organisational: Exec release readiness</td>
<td></td>
<td>Software product engineering Organisation process definition</td>
</tr>
<tr>
<td>Management: Project management plan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Management: Project planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support: Process improvements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Management: Risk management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organisational: Exec test plan review</td>
<td>4.3: Contract review</td>
<td>Requirements management Software project planning Software subcontract management</td>
</tr>
<tr>
<td>Organisational: Project approval request</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organisational: Exec test plan review</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organisational: Exec release readiness</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support: Process improvements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Management: Development process</td>
<td>4.4: Design control</td>
<td>Software project planning Software project tracking and oversight Software product engineering Software quality management</td>
</tr>
<tr>
<td>Management: Project review</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support: Configuration management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support: Document management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support: Quality assurance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support: Verification</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support: Validation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support: Process Improvement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organisational: Requirements Def</td>
<td>4.5: Document and data control</td>
<td>Software configuration management Software product engineering</td>
</tr>
<tr>
<td>Management: Requirements capture</td>
<td></td>
<td>Software subcontract management</td>
</tr>
<tr>
<td>Support: Third party software &amp; reuse implementation: Approach definition implementation: Software maintenance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support: Configuration management</td>
<td>4.6: Purchasing</td>
<td>Software subcontract management</td>
</tr>
<tr>
<td>Support: Document management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support: Verification</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support: Validation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support: Process Improvement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Management: Requirements capture</td>
<td>4.7: Control of customer-supplied product</td>
<td>Software subcontract management</td>
</tr>
<tr>
<td>Management: Monitoring &amp; control implementation: Approach definition implementation: Software maintenance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Management: Development process</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support: Change control implementation: Software design implementation: Software construction implementation: System integration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support: Configuration management</td>
<td>4.8: Product identification and trace</td>
<td>Software configuration management Software subcontract management</td>
</tr>
<tr>
<td>Support: Document management</td>
<td></td>
<td>Software product engineering</td>
</tr>
<tr>
<td>Support: Process Improvement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Management: Requirements capture</td>
<td>4.9: Process control</td>
<td>Software subcontract management</td>
</tr>
<tr>
<td>Support: Third party software &amp; reuse implementation: Approach definition implementation: Software maintenance</td>
<td></td>
<td>Software product engineering Software quality assurance</td>
</tr>
<tr>
<td>Support: Configuration management</td>
<td></td>
<td>Software product engineering Quantitative process management</td>
</tr>
<tr>
<td>Support: Process Improvement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organisational: Exec test plan review</td>
<td>4.10: Inspection and testing</td>
<td>Software product engineering</td>
</tr>
<tr>
<td>Management: Release strategy</td>
<td></td>
<td>Peer reviews</td>
</tr>
<tr>
<td>Management: Test management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support: Configuration management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support: Process improvement</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Appendix I – QFSD Process contributor relevance questionnaire

Table 11: QFSD Process contributor relevance questionnaire

<table>
<thead>
<tr>
<th>QFSD Process contributor relevance questionnaire</th>
<th>Interested Group</th>
<th>Relevance Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organisational Processes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Company Objectives</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business Case</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Objectives</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Approval Request</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements Definition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Executive Test Plan Review</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Executive Release Readiness</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Improvements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Management Processes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Management Plan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release Strategy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resource Strategy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements Capture</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Development Process</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Risk Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monitoring &amp; Control</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Review</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem Resolution</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support Processes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configuration Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quality Assurance</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Verification (inspection)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Third Party Software &amp; Reuse</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defect Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Improvement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change Control</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation Processes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approach Definition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements Definition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Design</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Construction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defect Repair Strategy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Integration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release Testing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organisational Readiness</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Maintenance</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table J1: QFSD User survey questionnaire

<table>
<thead>
<tr>
<th>Survey Question</th>
<th>Response</th>
<th>Name</th>
<th>Department</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is your job role?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provide a brief description of job responsibilities</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of projects in which online QFSD framework was applied?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>What benefits did you observe from the application of the framework?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>What benefits did the framework provide in terms of your job role?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>How could the framework be improved?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
24 Appendix K – QFD matrix overview
Software QFD matrix overview

Company
Business
Objectives
Matrix

Product
Marketing / User
Objectives
Matrix

Development Teams
Objectives
Matrix

Prioritised Tasks
Matrix

Detailed Tasks
Matrix

Project Planning
Requirements questionnaire set to beneficiaries.

<table>
<thead>
<tr>
<th>Req</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req</td>
<td>2</td>
</tr>
<tr>
<td>Req</td>
<td>3</td>
</tr>
<tr>
<td>Req</td>
<td>4</td>
</tr>
<tr>
<td>Req</td>
<td>N</td>
</tr>
</tbody>
</table>
26 Appendix M – Fisher-Rosemount Case Study Three

26.1 Application of QFSD core processes in Case Study Three
The following sections apply the application of each of the QFSD processes to the project planning and execution tasks and describes the unique benefits gained.

26.1.1 Organisational processes

26.1.1.1 Company objectives
The first step in the project is to understand which of the company's objectives this project would help to achieve. The following are the major company objectives identified:

- Reduce customers cost of ownership of their process control systems.
- Move the company's current process control system product range on to a wider range of industry standard server platforms, thereby generating cost savings for customers in terms of providing low cost PC clients whilst also enabling scalability and choice in the server platforms.
- Improve profit margins on system engineering contracts.
- Sustain sales of the existing product range until the full Intel-based replacement control system products are ready to deliver.

26.1.1.1.1 Benefits
Aligning a project's objectives formally with the main company objectives may seem to be an obvious step. However, consider a normal IT budget planning exercise. Typically an IT department's revenue budget consists of a high level of maintenance staff costs (a third in the case of Celesio), an area called business-as-usual projects (a third in the case of Celesio) and a limited number of special strategic projects or programmes (again, a third of the budget in the case of Celesio). Whilst the strategic projects are likely to be aligned with the objectives of the business and have a high profile within the business, the remaining two thirds of the budget are largely directed by the IT department itself. Business-as-usual projects are typically undefined at budget time and are based on the previous years efforts involved in doing ad-hoc business change requests and projects that
the IT department believe are necessary in order to keep the business applications running, improve communications, improve security, reduce costs etc. However, this is a major portion of the IT investment that is not checked for business alignment and can result in projects not being seen as successful, projects being stopped, and the IT department having to account for low delivery and a relatively high level of spending. This could lead to the outsourcing of the IT function due to the business believing that they will get better service and cost transparency, which in many instances results in increased costs as an outsource partner is more than happy to be transparent in terms of billed services. A number of major companies have been through the outsource route only to revert to in-house services due to high costs (Baitheiemy, J, 2003).

26.1.1.2 Business Case

The researcher contributed to the preparation of a business case for the Case Study was prepared in conjunction with the product marketing department and representatives from the main target users of the new system. This enabled some very early validation that the right product was to be delivered.

The business case was initially based on the projects cost savings and increased revenue projections for the first year of release. The elements of the business case were as follows:

- Savings made by the Fisher-Rosemount Company's internal engineering teams by increasing engineering contract margins.
- Savings made by the company's value added resellers by enabling them to increasing their engineering contract margins.
- Reduction in the cost of the minimum 'foot-print' process control system.
- Company development cost reductions due to consolidation on one platform.
- Increased generation of new control system sales based on the ability to bid for smaller control system business.
- Reduction in technical support costs as only one code line is supported.
26.1.1.2.1 Benefits

The business case is the most important processes required by the QFSD framework or indeed any software development approach. Not only does a software deliverable need to contribute to the company’s business objectives it also needs to generate a return on the investment within a defined timescale. Again this would seem like an obvious statement, but the researcher has observed a significant number of projects that either do not have a proper return on investment calculation or have such a calculation at the initial budget approval stage only to find that the actual cost of the project is much higher once the design stage is completed and hence never attains the agreed ROI.

26.1.1.3 Project approval request (PAR)

The PAR is the official document, which grants the project the right to expend company funds in the creation of software, and the ultimate release of software deliverables.

In order to generate the PAR the following processes needed to be completed:

• Project concept document / prototype and customer validation.
• Project objectives
• Release strategy
• Resource strategy
• Risk management strategy
• Initial capital and lifecycle costs
• Process improvements

In order to help the reader in understand the scope of the case study project, the project objectives section from the case study PAR document has been included as follows:

The case study project objectives are:

Using the six business case objectives listed in Section 7.2.1.2 the case study project goals are defined as:
• Demonstrate to customers by the delivery of the new open platform
  configuration product, that the current process control product range was still a
  viable option for new and existing customers.
• Provide a fully integrated open platform replacement for the current workstation
  configuration products and therefore reduce future maintenance and
  development costs.
• Address the issue of 'ease of configuration' that has been levelled against the
  existing workstation product.
• Significantly reduce the amount of effort to configure the process control
  systems, thereby improving system engineering contract margins.
• Migration of the installed base custom configuration databases gracefully from
  the multi-platform workstation environment to the new open platforms.

26.1.1.3.1 Benefits
The PAR provides a mechanism to gain approval from all responsible parties for
the software project to start expending company funds. Approval from all project
sponsors is vital in order that joint responsibility for the project is established at the
outset. This approach ensures that all departments in an organisation are aware
of the project, its scope, budget, timeline and their role in its successful delivery.
Information from the PAR document can also be used as the source information
for a company project portfolio management process.

26.1.1.4 Project management plan
The PAR is really the starting point of the project management plan and
addresses the projects business case.

The detailed project processes were defined and recorded in the projects project
management plan. The contents of the project management plan reflected the
structure as shown in Appendix E.
26.1.1.4.1 Benefits

As the project management plan is the focus of all project planning and control, it is necessary to make its contents more accessible to the project group, external departments and senior management.

In addition, the project management plan refers to other project information such as schedules, task lists, test logs etc. The dynamic nature of this information and its diversity needed to be addressed.

Whilst all processes are defined in the online repository along with their associated document templates, the project management plan is handled in a slight different way. Rather than it being a document located within the projects online repository folders, it was provided as a set of HTML pages, each page covering a different area of the project management plan with a content management application provided in order that the project management plan could be updated online and immediately made available to all project members. The approach taken was to first convert each major section of the original project management plan template (organisational processes, management processes, support processes and execution processes) in to HTML pages. These pages were then included in the intranet based QFSD online repository again as a template. All project members and sponsors have access to the online repository and hence have access to the latest project management plan information. An alert mechanism alerts them to the fact that the project management plan has changed and what pages should be accessed.

26.1.1.5 Requirements Definition

The requirements gathering and definition process is based on a modified version of the quality function deployment (QFD) methodology as described in Chapter 6.

This approach proved to be very successful and avoided the late requirements and requirements creep suffered in the first case study.

26.1.1.5.1 Benefits

The benefit of having a reliable requirements gathering and definition process in this case study was clear. Whilst the project had defined its objectives within the PAR document, a number of the objectives could have generated an endless wish
list of requirements. In reality, the formality of the modified QFD requirements methodology enabled an achievable set of requirements to be generated and agreed with all stakeholders.

For a detailed example of the methodology used for this case study refer to the QFD section in Chapter 6.

26.1.1.6 Executive test plan review

This meeting is held early on in the project lifecycle as soon as the top-level test plan is produced. The reason why this process was created was due to the researcher's experience at Fisher-Rosemount of delivering projects on-time, but having the released delayed by senior management who insisted on questioning all aspects of the project and seemed to be nervous of releasing new products to their customers. Given the experiences they had with the project in the first case study, it is understandable why this was the case. Therefore, in order that senior management were involved in the project at any early stage and had input in to the user testing, pilot and release strategy this new process was added to the framework.

26.1.1.6.1 Benefits

Firstly, the review ensures that the type and level of testing is appropriate for the software product. Also, by having the senior managers from the technology, marketing and operations groups present at the review, a certain level of joint ownership is fostered. This joint ownership is needed in the later project stages when resources and assistance are required from the other groups. Whilst the executive test plan covers all aspects of testing, the key area is in jointly setting the release criteria. Chapter 6 describes how defect discovery rate statistical analysis can be used for determining when a software product is ready to be released in to a given market.

26.1.1.7 Executive release readiness review

This meeting is held at the end of release testing and field trials. The objectives of the meeting are as follows:

- To determine if the software product is ready for release
• To ensure that the organization as a whole is ready for the release and able to provide manufacturing and technical support

• To establish joint responsibility for the release. i.e. a collective decision by all groups is taken and recorded

The Executive Release Readiness Review presentation format is defined in a top-level company procedure, CP1101.

26.1.1.7.1 Benefits

This process is critical to ensure that the organisation is aware that the software product is to be released and as a final check that the organisation is prepared. Organisational readiness is a key part of ensuring that the software product attains its stated ROI. For example, before the product in the case study was released the organisational readiness process ensured that the following groups were ready to support the release:

• Software manufacturing in order that they can produce the media and manuals to meet demand

• Technical support in order that the product and any customer migrations can be fully supported

• Internal and external engineering groups such that they are aware of the new capabilities

• Those engineering groups who participated in the pre-release trials who were given the immediate release of the new software product

• Other software product development groups who received communication of the release as the case study project was part of a co-ordinated Control System release involving a new control room operator console and new field controllers

Failure to carry out any of the above will result in a suboptimal release and can have a negative impact on co-operation with other groups within the organisation for future releases.
26.1.1.8  Process improvements

The process improvement assessment is a key activity that is required in order to continuously improve the QFSD framework and hence the overall software development quality level. Unlike other framework methodologies, QFSD builds in continuous process performance and process capability monitoring as part of each software project. Please refer to Chapter 6 for a detailed description of the process performance and capability tools and associated techniques. The project post-mortems of previous projects should also provide a rich source of process improvement ideas. In addition, if the QFSD model is being enhanced to use all, or part, of a new technique, then part of the introduction plan should be factored in to the project as a process improvement. Process improvement is therefore treated in the same way as any other project requirement, with the same level of importance as a priority one, business requirement. In this way, process improvement becomes the responsibility of all members of the project team and not a single quality assurance person.

The process improvements included in the case study project were as follows:

- A process oriented structure to the project management plan
- A new document numbering database user interface
- An improved online help system design, having technical writers work in the development team rather than subcontracting the work out to a specialist company
- Full adoption of the new user requirement capture method (used a modified version of Quality Function Deployment [QFD]).
- Introduction of the defect / hour metric to monitor the stability of the product during release testing and set defect rate criteria for the release.

26.1.1.8.1  Benefits

The QFSD framework places a high priority on continuous process improvement and supports this with the necessary processes and tools. By taking this approach
the QFSD framework has evolved rapidly and ensures that quality is an integral part of the software development process and not an arbitrary hurdle set by the quality department.

26.1.2 Management processes

26.1.2.1 Project planning

A project must be planned to succeed. The three keys were identified for the successful delivery of a software project:

- Establish an achievable schedule (May, 1998)
- Manage requirements scope (Leffingwell and Widrig, 1999)
- Deliver validated requirements (Kotonya and Sommerville, 1998)

Establishing an achievable schedule depends to a degree on the more obvious factors such as resource availability, skill levels, accurate estimating, good project control and monitoring. However, by far the biggest danger is trying to meet every stakeholder perceived need. The pressure on the project manager to agree to a certain scope / time scale, before even the first set of developer estimates have been made or a set of requirements generated, is immense. This pressure comes from senior management, marketing and customers. There are two effective ways to deal with this pressure in order to avoid compromising quality levels in the delivered product.

The first approach is to give a range of dates (best case / worse case) and refuse to give a more concrete schedule until the development team has fully understood the requirements and the planning phase is completed. In new product developments, this needs to be extended to include the development and validation of a software architecture and creation of a performance model. The researcher has observed two major new software developments in Celesio AG that have failed to validate their software architectures and both have suffered in terms of significant schedule delays and increased costs.

This first approach may not be that successful, therefore, if the project is forced into giving dates, then the second approach is that only features with high priorities should be committed. The requirements capture and prioritisation technique
described in Chapter 6 was designed to enable the project team to manage the scope by involving stakeholders at an early stage in a detailed requirements prioritisation process. Having a set end date forced the project to prioritise the features in to three categories, only guaranteeing to deliver on the priority one items. This approach was balanced to ensure that the delivery of the priority one items alone would meet the basic requirements of the business case.

26.1.2.1.1 Benefits

Management of requirement scope is very difficult. Late requirements come from many sources and are presented to the project manager with a claim that the whole product will fail if the requirement is not included. This is where having a strong business case comes in to its own. The additional late requirements can be weighed against the business objectives and will then stand or fall on their individual merits. All project requirements must have been first validated against the business needs of the company and its customers. If a product does not satisfy both these stakeholder groups, then it may never be classified as a commercial success, even if it is a technical triumph. The QFSD framework provides both the QFD requirements definition process and the clusterisation development techniques in order to support the three key planning rules listed in the previous section.

26.1.2.2 Planning & estimating approach used.

This section describes how the QFD and clusterisation techniques were used as the basis of the case study planning process. Please refer to the Figure M1 'Project Planning Process', which gives an overview of the process, used to plan this project. The results of the QFD analysis for each major project area were extracted from the QFD spreadsheet and recorded in the initial planning spreadsheet. The project development and delivery approach was then introduced to the project team. In this case study, the project worked towards four major milestones. The first three were the delivery of partially complete, but fully operational functional areas i.e. totally working features were delivered for customer validation at the end of each milestone. The final milestone was the delivery of a fully tested software product ready for field trial and subsequently software manufacture and customer delivery. The milestones were assigned dates
and entered as columns in the planning spreadsheet. In addition, further columns were added to record effort to implement, cost and resource assignment. The cluster leaders, together with the project manager then considered each requirement and assigned the following:

- The milestone in which the requirement was to be implemented and delivered.
- Effort to implement in terms of person years and actual cost. (Person years were used initially as the estimates could only be based on gut feel using historic information and experience due to this being a new product)
- Resources concerning who was best suited to implementation each particular requirement

The cluster leaders and project manager also discusses the requirements for services between project clusters. For example, they noted that the user interface cluster may require buffering capabilities in the middle tier of the product. These inter-cluster dependencies are a useful indicator of the projects critical path implementation items.

At this point the planning spreadsheet had all the requirements assigned to milestones, first pass estimates and initial resource assignments. The project manager then added those generic project tasks, which are best estimated based on previous historic project information. Those generic project tasks include:

- Inspections
- Defect removal
- Project administration (meetings, training etc).
- Project management
- Test plan production
- Test execution
- Requirements capture
- Development system support
Using the information from the initial planning spreadsheet the project manager and cluster leaders built the first project schedule. The schedule showed the order in which the main project tasks would be executed, which cluster project groups needed to exist and the basic resource profiles. In parallel with the above activities the project teams requirements capture group, started to write the draft software analysis & requirements specifications for each of the main clusters (Note that each project cluster was responsible for a major feature or set of features).

Following the first reviews of the draft analysis & software requirement specifications, the cluster groups produced a set of task lists (one set per cluster group). These task lists were populated with the first pass detailed estimates. This process was carried out on a per milestone basis i.e. detailed tasks and estimates are produced for milestone 1 and at the end of milestone 1 for milestone 2 and so forth. This ensured that the detailed estimates for each milestone were produced when the developers were fully aware of the details of the tasks they are required to carry out. It also enabled them to gain implementation experience using the new technologies from the previous milestone, which made their estimates increasingly more accurate.

The project manager and cluster leaders then reviewed the first pass task lists and estimates. This review considered the validity of the estimates, skills available to execute the tasks, priority assigned from QFD analysis and schedule time available. Based on the review, a second set of tasks / estimates were generated by the cluster groups. In addition, a second schedule was produced. At this stage the cluster leaders were asked to identify the inter-cluster dependencies. This information is vital from a planning point of view. If the dependencies are not identified in the planning stage, then clusters will be waiting for services from other clusters, thus giving them dead time.

A second meeting took place to review the tasks / estimates / priorities and inter cluster dependencies. The results of this meeting were used to set up the definitive project task tracking system and project schedule and were also used as the basis of minor adjustments to the software requirement specifications before being finally inspected and approved.
The estimates and schedule were then base-lined in the project development folder (which is the QFSD online repository), for future project planning reference and comparison with actual performance at the end of the project.

![Diagram of Project Planning Process]

**Figure M1: The Project Planning Process.**

### 26.1.2.2.1 Benefits

The planning process in terms of software development is clearly necessary. Again the QFSD framework takes this in to account and is somewhat prescriptive, which is not the case with other frameworks. Clearly any effective planning approach can be used with the framework, but in the case where a development group does not have a tried and tested approach, the framework provides a proven process.

### 26.1.2.3 Release strategy

Based on the hard lessons learned in the first case study, it was clear that attempting a total replacement for the existing configuration tool, using a new development language and operating system, presented an unacceptable risk to the company. The strategy was to gradually phase out the existing workstation...
based multi-platform configuration product, whilst incrementally replacing its functionality over three releases. In addition, the customers existing configuration data, in which the engineering investment runs in to millions of pounds, would be preserved.

This strategy would be achieved by the following:

1. Replace the old systems client functionality with a new PC client release, which connects to the same database platform as the existing workstation clients. In this way, the new product can be added as an additional client to existing customer systems.

2. Collect customer feedback on the first client product release for inclusion in a second client. Include features not provided in the first client release.

3. Move the database and server applications on to the Windows platform and also have a POSIX server option. This will be the third release of the new product entirely on the Windows platform. At this point, the old VAX based workstation product will be phased out. Again, customer databases will be preserved, as the same database engine will be used on the Windows and POSIX platforms, enabling a simple data transfer to take place.

26.1.2.3.1 Benefits

A release strategy is important both in terms of release of product to the customer base and in terms of how it is released and supported by the organisation. Again the QFSD framework recognises that the release strategy can be a major contributory fact to the overall success of a software product.

26.1.2.4 Resource Strategy

In practice, a development department generally has a team with mixed skill levels and in some cases a shortage of specific skills. This case study project was no exception and resource planning needed to be considered carefully. The resource loading shown in Figure M2 for the project shows the resource profile of the project team progressively building up to ten during the first seven months of the project. This low number was not due to staff shortages, but because the design, programming language used and project build systems all needed to be mastered
before adding more resources. These first ten developers were the most highly skilled, with three of them being specialised contractors.

Once the project team had the overall design in place, understood the new programming language, built the basic object framework, and established a build system, the team was expanded in order to complete the bulk of the coding and release testing.

The resource profile in Figure M2 shows how resources were added gradually to the project. Notice that the resources taper off towards the end of the project. This is in contrast to the staff profile in the first case study, where resources were at their maximum at the point of release.

Applying too many resources at the start of a project, is just as bad as applying them towards the end of a project that is running late. Front loading will cause the project to "spin its wheels" as team leaders struggle to understand requirements, new technology and organise teams. Back end loading, other than for bulk user testing, will cause the project to slow down as existing resources train the new starters. Adding more resources means more work can be done, but it usually increases schedule by a factor greater than the ratio of existing resources to new ones added.

![Figure M2: Second case study staff profile](image)
26.1.2.4.1 Benefits

Ensuring that a software project has both the correct numbers of staff with the required skill levels is always a challenge, especially when the project is completing for key resources within a department's overall resource pool. The clusterisation approach helps with this issue by ensuring that developers are progressively trained in all aspects of the department technologies in order that they can contribute to any area within development.

26.1.2.5 Requirements capture

This is the process by which the requirements were established using the QFD approach as described in Chapter 6. Each major development area was assigned a specific software requirement specification. These areas were mapped on to the project development clusters during the QFD analysis. The requirement specifications contained unambiguous statements of requirement. Each requirement was given a unique requirement number, which enabled it to be traced through the coding and testing phases. Each software requirements specification was subjected to a development team inspection and a stakeholder inspection using the inspection process as described in Chapter 6.

26.1.2.5.1 Benefits

Rather than simply stating that requirements capture needs to be carried out the QFSD framework provides techniques to support the prioritisation, tracking, documenting, formal inspection and configuration management. Again the QFSD framework provides a combination of quality process description and a practical technique and associated tool-set for implementing the technique.

26.1.2.6 Development Process

The QFSD framework recommends an incremental approach to development rather than an iterative approach. However, each increment has a percentage
estimate for rework and hence has a controlled element of incremental rework built in.

In a later chapter the RUP, which is iterative in nature, is provided as an option within the framework. Whilst a purely iterative approach should perhaps only be used by very experienced development teams with strong requirements management processes in place, it is an accepted development approach and hence the framework provides it as an alternative.

This project used the clusterisation approach as the basis of its development process. There were four cluster groups established to create the fundamental architecture components and base features. The four clusters were: Server Engineering, User Interface, CDOS and Snap-On. The Server Engineering group developed a POSIX version of the server database and back-end processing that would run on both UNIX and Windows operating systems. The User Interface cluster developed the new configuration interface e.g. basic configuration screens, spreadsheet screens for rapid data entry, drag 'n' drop features, graphical query and reporting etc. The CDOS group provided the connectivity between the new PC based clients and the existing and new workstation based database servers. The Snap-On group-produced value adds graphics and HTML applications, which would be licensed add-ons to the base product for bulk data configuration entry.

Later three further clusters were established, one to implement the enhancements, one to carry out defect repairs and an independent test team cluster. The first four cluster groups worked concurrently as did two of the last three clusters. However, each cluster can be in a different development phase i.e. design, code or unit test, which makes the approach partially iterative.

Each cluster was responsible for the delivery of its own feature list, which includes estimation, design, code, unit test, and integration with other clusters (remember the task definition process required the inter cluster dependencies to be identified) and each cluster was responsible for the quality of the deliverables. Each cluster group produced its own detailed task list and associated estimates broken down by phase. This information was used in the main project tracking system, with the cluster leader being responsible for updating their cluster's actual effort expended, re-estimates, and new task additions on a weekly basis. This approach ensures
that the cluster groups did actually plan the whole cluster execution tasks, and had considered the design and required resources.

The project was divided into six milestones. Each milestone was of the order of 6 to 8 weeks in duration. At each milestone, the clusters were required to deliver a planned set of functionality. Following the milestone a period of integration took place with a week's testing. This approach ensured that at the end of each milestone the project had a set of installable software executables, which could be handed off to external groups for validation and early testing. This approach provided to be very successful as the configuration product was required as part of the field controllers and control room console testing phases. At the end of each milestone, the project was reviewed in terms of performance and process success. This approach also enabled certain processes to be improved during the project and provided feedback on the previous milestones estimates, which allowed the subsequent milestone estimates to be continuously refined. Since this was the first release of a new product, using an unfamiliar programming language and operating system, this approach enabled a continuous reassessment of which features could be delivered in a given milestone.

Table M1 shows how the project estimate accuracy improved as the project reached each milestone.

Table M1: Estimate refinement technique

<table>
<thead>
<tr>
<th>Project Areas</th>
<th>Actual Effort</th>
<th>Milestone One</th>
<th>Milestone Two</th>
<th>Milestone Three</th>
<th>Milestone Four</th>
<th>Milestone Five</th>
<th>Milestone Six</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management</td>
<td>3655 hr</td>
<td>2000 hr</td>
<td>2000 hr</td>
<td>2000 hr</td>
<td>3000 hr</td>
<td>3250 hr</td>
<td>3300 hr</td>
</tr>
<tr>
<td>Inspections</td>
<td>1878 hr</td>
<td>1500 hr</td>
<td>1500 hr</td>
<td>1500 hr</td>
<td>1600 hr</td>
<td>1700 hr</td>
<td>1800 hr</td>
</tr>
<tr>
<td>Training</td>
<td>1137 hr</td>
<td>800 hr</td>
<td>800 hr</td>
<td>900 hr</td>
<td>1100 hr</td>
<td>1100 hr</td>
<td>1200 hr</td>
</tr>
<tr>
<td>Increment 1</td>
<td>2074 hr</td>
<td>2074 hr</td>
<td>2074 hr</td>
<td>2074 hr</td>
<td>2074 hr</td>
<td>2074 hr</td>
<td>2074 hr</td>
</tr>
<tr>
<td>Increment 2</td>
<td>1771 hr</td>
<td>2000 hr</td>
<td>2000 hr</td>
<td>1771 hr</td>
<td>1771 hr</td>
<td>1771 hr</td>
<td>1771 hr</td>
</tr>
<tr>
<td>Increment 3</td>
<td>2340 hr</td>
<td>2000 hr</td>
<td>2000 hr</td>
<td>2340 hr</td>
<td>2340 hr</td>
<td>2340 hr</td>
<td>2340 hr</td>
</tr>
<tr>
<td>Increment 4</td>
<td>2907 hr</td>
<td>2300 hr</td>
<td>2300 hr</td>
<td>2907 hr</td>
<td>2907 hr</td>
<td>2907 hr</td>
<td>2907 hr</td>
</tr>
<tr>
<td>Increment 5</td>
<td>1375 hr</td>
<td>1800 hr</td>
<td>1800 hr</td>
<td>1900 hr</td>
<td>1900 hr</td>
<td>1375 hr</td>
<td>1375 hr</td>
</tr>
<tr>
<td>Increment 6</td>
<td>6044 hr</td>
<td>4000 hr</td>
<td>4000 hr</td>
<td>4500 hr</td>
<td>4900 hr</td>
<td>5400 hr</td>
<td>5500 hr</td>
</tr>
<tr>
<td>Totals</td>
<td>22348 hr</td>
<td>18700 hr</td>
<td>18472 hr</td>
<td>19485 hr</td>
<td>21592 hr</td>
<td>21917 hr</td>
<td>22267 hr</td>
</tr>
<tr>
<td>% Accuracy</td>
<td>75%</td>
<td>79.5%</td>
<td>84%</td>
<td>93.9%</td>
<td>94.3%</td>
<td>96%</td>
<td>96%</td>
</tr>
</tbody>
</table>

The table shows that even with detailed estimating based on historic data, the initial estimates for the project were 25% under the actual effort used. As more
work was carried out on the project and experience grew, then estimates to complete become progressively more accurate.

The researcher has seen this under estimating consistently in software projects and has observed companies who double their development team’s estimates as a rule of thumb. However, as a rule of thumb, it is better to add 25% (Molokken and Jorgensen, 2003) as a contingency for software projects that contain a large proportion of new technologies, as estimating is usually inaccurate as unforeseen problems do occur in most software projects and it makes good sense to have planned capacity to deal with them.

It could be said that over estimating is a way of being seen to hit schedule. This may be partially true, but consistent over estimating loses the confidence of the stakeholders and the perception that IT department deliberately over estimate in order to avoid delivery pressure.

26.1.2.6.1 Benefits

There are many development methods that can be employed in software development. The key is to match the development method with both the type of software development and the quality levels required. In this regard the QFSD framework can support a number of development techniques. However, the incremental clusterisation approach has been used by the researcher to successfully deliver both commercial and safety critical software applications and comes as the default method in the core framework.

26.1.2.7 Risk Management

Risk management is a key element in the management and execution of a software project. However, simply writing down a list of risks and trying to resolve them all is not efficient in terms of overall project effort, costs and schedules.

The QFSD process for risk management is based on a statistical probability approach. Details of the calculations used are detailed on Appendix C. The process establishes a dynamic list of potential project risks to be identified and the probability of their occurrence in the project. Each risk also has an estimate of effort and cost required to mitigate the risk. The effort and cost estimates are based on risk mitigation plans created previously for each identified risk.
established both the probability of a risk occurring and the cost of mitigation, the project manager can then make an informed decision on which risks to address and in what order.

The risks associated with the case study project were formally documented in the project management plan. However, a separate risk log was created in order that the risks and their associated management plan could be managed on a day-to-day basis. Once the major risks were identified, they were reviewed by both the project team and the main project stakeholders prioritised and classified. The standard classifications used were:

- Risk retention
- Risk avoidance
- Risk transfer
- Risk reduction

26.1.2.7.1 Benefits

Taking a probability based approach to risk management avoids the difficulty in allocating resources properly. Resources spent on risk management could be used on more profitable activities. The approach used in the QFSD framework spends the least amount of resources while reducing the impact of high probability risks.

26.1.2.8 Monitoring and control

The frequency and approach for project tracking depends to a certain extent on how the project tasks and milestones have been defined. Again the QFSD framework is prescriptive in this area in order to ensure that the correct level of monitoring is applied. Without appropriate and frequent monitoring a project can quickly go out of control.

For this project, three levels of project reporting were adopted:

1. A written monthly report, which was produced by the project manager and sent to the Fisher-Rosemount product range director. The product range director then produced a summary report to senior management containing
progress and risk issues associated with all products currently under enhancement and contributing to the next release of the product range.

2. A fortnightly project progress meeting at which the project manager reviews progress with the cluster leaders (minutes / actions are recorded in the QFSD online repository). Materials reviewed were then updated including monitoring spreadsheets and the project schedule. The project schedule is marked up at the meeting and made available to the team by publishing it in the QFSD online repository.

3. A series of team meetings, which took place on a need-to basis, or on planned milestone completion dates.

26.1.2.8.1 Benefits

Rather than simply stating that projects must have monitoring and control processes, but not really elaborating on them, the QFSD framework is quite prescriptive. For example, a typical recommendation from the framework taken from the case studies project management plan is the need to create an interdepartmental team (IDT). The IDT developed the product support, manufacturing, education, ordering and documentation plans. The IDT groups consisted of representatives from each of the major functions outside the IT group.

These included:

- Operations
- Manufacturing / North America / Europe and Asia Pacific
- Technical Documentation
- Product Marketing
- Drafting
- Purchasing
- Technology
As the project approached release, these IDT meetings become more frequent in order and made sure all departments were prepared and trained for the release of the software deliverable.

Whilst the above level of monitoring may be seen as a little excessive it was found in the case study, and in other projects, that higher visibility of the software production processes enabled corrective actions to be applied throughout the project, which made a major contribution to keeping the project on schedule.

26.1.2.9 Test Management

The recommendations and templates for testing within the framework are fairly standard and include such techniques as automated regression testing. All test artefacts were held in the QFSD online repository and all test cases were mapped to their corresponding requirements. However, the unique elements of the framework in terms of testing were applied with great success. The defect rate measurement techniques were used to determine when a software product was ready for release. The result was that overall test duration and effort were roughly comparable to previous testing approaches used. However, the areas in which testing was applied became much more focussed with a definite test completion driven by achieving the target defect rates.

26.1.2.9.1 Benefits

The QFSD framework ensures that a software product is tested thoroughly, but also that it is not over tested or that tests are run in areas of the product that are already stable. Using the defect rate approach, as described in Chapter 6, ensured that the software product was sufficiently stable to be released and that over testing is not carried out. A follow-up analysis post-release indicated that the defects found during the first six months of deployment were 34% fewer than previous releases. However, it is likely that a proportion of this improvement is due to improved processes applied across the board from the QFSD framework.