The development and use of a toolset for industrial IT portfolio management

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 Doctor of Philosophy of Loughborough University.

Metadata Record: [https://dspace.lboro.ac.uk/2134/10695](https://dspace.lboro.ac.uk/2134/10695)

Publisher: © Mark De Chazal

Please cite the published version.
This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/
Please note that fines are charged on ALL overdue items.

FOR REFERENCE ONLY
The Development and Use of a Toolset for Industrial IT Portfolio Management

by

Mark de Chazal

Supervisor: R. J. Dawson
Acknowledgements

I would like to thank several people, whose help and assistance made this thesis possible:

I would like to thank Bill Simmons, for authorising the sponsorship, and providing the environment in which I could work. I would also like to thank Malcolm Bradley, without whom I think this thesis would never have been. I have accumulated many pearls of wisdom from sharing an office with him, including the contents of the “MBy Book on Tact & Diplomacy!”.

Heulwen Pearce and Steve Brotherhood have been supportive throughout, giving freely of their time, helping an IT-trained economist understand an engineering company. They have both helped me tremendously – I wouldn’t be where I am today without the knowledge they instilled in me! Thank you!

Special praise must go to Ray Dawson, who has been unflagging in his support, and the best supervisor I have ever had!

Finally, this thesis has required me to work on-site at Rolls-Royce and work within several teams. This thesis would not have been possible if the team members had not been willing to accept a PhD student as a legitimate member of the teams. I would like to thank (in no particular order):

Lian Blackshaw, Sarah ‘Saz’ Robinson, Kirsty Adamson, Phil Bath, Judith Connolly, Liz Hardwick, Ellen Grundy, Adrian ‘Happy’ Gailey, Nigel ‘Sam’ Salt, Joe Levy, Tom Cosgrove, Jo Hughes, Liz Hookey, Anne Houston, Lisa Collins, Mark Baker, Mike Larimore, Stu Bull, Jean Cadas, Paul Collinge, Tony Brett, Steve Smalley, Jez Sear, Stuart Wicks, Veronica Jurrison, Sally Ainley, Annette Grassi, Charlie Oldham, James White, John Newton, Steph Hayes, Marcus Sample, Alex Helm, Kevin Watson, Claire Turner, Mark Goodson, Kevin Watson, Sue Hadson, Ian Newman, Tom Jackson, Irene O’Malley, Madeline ‘Mad’ Emuss, Claire and Dennis at the Front Gate, and lastly but not least, my family (including miscellaneous grandparents, aunts, uncles, and cousins)

All of the above have helped me tremendously in either allowing me to work, helping me to work, or stopping me working too much!
Abstract

Many companies have had adverse experiences of information system costing more and delivering less than originally planned. This thesis proposes a tool set for effective IT portfolio management. Rolls-Royce Naval Marine sponsored a study to create, develop and evaluate a practical IT portfolio management toolset. One implication of this was that the study took place entirely within a 'live' environment – the tools were developed in close partnership with the employees of Rolls-Royce, and the work performed was a significant contributor to the profit margin of Rolls-Royce Naval Marine Support. The initial version of the toolset was piloted in the Configuration Management area of the Submarine Support business units. Other Rolls-Royce business units then provided a test for the general applicability of the methodologies and tools developed. The objectives of the IT portfolio management strategy were to maintain current business capability, enable future business, and where possible, identify possible cost savings through better usage of information.

The portfolio management framework drew upon Checkland's soft systems methodology, extensively adapting it to suit the specific situation faced. Various tools were employed to solicit current and future business requirements. Feature analysis was evaluated for use in matching features to systems, cost, strategic intent, and process. Diagrammatic tools included different types of rich pictures, entity relationship diagrams and contextual data flow diagrams. Gap analysis illustrated the difference between capability and requirements. Techniques were developed to portray the results so that they could be effectively communicated. Options were evaluated using various methods for assessing cost-effectiveness. A technique was developed to manage the various stakeholders involved in the information systems strategy.

The output was a cost-effective, implementable strategy that supported both current and future business requirements, and had buy-in from users, developers, and management. The IT portfolio management strategy was evaluated by identifying actual and potential operational savings, leading to a rationalisation of the methodology to provide more flexibility.

The IT portfolio management strategy developed is specific to the business site. However, the process by which it was created can be used by any business, and is very flexible. Several of the tools have been used in other parts of Rolls-Royce with success, two of them are under consideration for adoption into Rolls-Royce processes for global deployment. IT portfolio management in other areas of Rolls-Royce is being increasingly influenced by the framework and toolset created during this thesis.
# Table of Contents

Acknowledgements .................................................................................................................. 2  
Abstract .................................................................................................................................. 3  
Table of Contents ...................................................................................................................... 4  
Table of Figures ........................................................................................................................ 9  
Table of Tables .......................................................................................................................... 11  
Chapter 1: Introduction .............................................................................................................. 13  
1.1 Introduction .......................................................................................................................... 13  
1.2 The Research Goal ................................................................................................................ 13  
1.3 Rolls-Royce – The Company .............................................................................................. 13  
1.4 Rolls-Royce Marine ............................................................................................................. 14  
1.5 Rolls-Royce Culture ........................................................................................................... 15  
1.6 Research Rationale ............................................................................................................. 17  
1.7 Aim ...................................................................................................................................... 19  
1.8 Objectives ............................................................................................................................ 19  
1.9 Thesis Outline ...................................................................................................................... 20  
1.10 Summary ............................................................................................................................ 21  
Chapter 2: Literature Review ...................................................................................................... 23  
2.1 Introduction ........................................................................................................................... 23  
2.2 IT System Portfolio Management ........................................................................................ 23  
2.3 Introduction to Legacy System – Causes and Definitions ................................................... 27  
2.4 System Evolution ................................................................................................................ 28  
2.5 Legacy System Definitions .................................................................................................. 32  
2.6 Maintenance Aspect of Legacy Systems .......................................................................... 37  
2.7 Quality Standards and Measures within the Business ....................................................... 39  
2.8 Soft Systems Methodologies ............................................................................................. 43  
2.9 Measuring Quality .............................................................................................................. 45  
2.10 Review of the Aim ............................................................................................................ 49  
2.11 Summary ............................................................................................................................ 49  
Chapter 3: Methodological Review ............................................................................................. 55  
3.1 Introduction ........................................................................................................................... 55  
3.2 Research Philosophy .......................................................................................................... 55
Table of Contents

3.3 Choice of Research Approach .................................................. 57
3.4 Research Approach ............................................................... 58
3.5 Research Strategy ................................................................. 59
3.6 Conducting the Research – The Choice of Research Strategy and Approach 71
3.7 Definition of the Stages ............................................................ 72
3.8 Summary ............................................................................. 77

Chapter 4: Period One – Introduction to the Workplace .................. 64
4.1 Introduction ......................................................................... 64
4.2 Overview ............................................................................ 65
4.3 Situational (Background) Research ........................................ 65
4.4 ‘Scare Story’ Development ..................................................... 65
4.5 Company Research ............................................................... 71
4.6 Lifecycle Costing .................................................................. 71
4.7 Expert Judgement .................................................................. 75
4.8 Feature Analysis .................................................................... 79
4.9 Statistical Analysis of Failure Data from IT Systems ............... 85
4.10 Consensus Building ............................................................... 85
4.11 Identification of Legacy Information Systems........................ 87
4.12 Research of Methods to Mitigate Legacy Information Systems 88
4.13 Summary ............................................................................ 100

Chapter 5: Period Two – Establishing Professional Credibility ........ 107
5.1 Introduction ......................................................................... 107
5.2 Overview ............................................................................ 107
5.3 Business Analysis Skills ......................................................... 108
5.4 Application of a Knowledge Base .......................................... 109
5.5 Modified Feature Analysis ...................................................... 114
5.6 Requirements Capture and Engineering .................................. 131
5.7 Alternative Methods to Analyse Defect Data ........................... 134
5.8 Graphical Defect Analysis ....................................................... 137
5.9 Statistical Analysis of Failure Data from IT Systems ............... 142
5.10 Questionnaire Use ............................................................... 143
5.11 Cultural Problems with Change ............................................ 148
5.12 Software Quality Models and Cost Models ......................... 148
5.13 Problem: Managing Stakeholders ......................................... 149
Table of Contents

5.14 Definition of 2nd Generation Problems ................................................. 149
5.15 Summary ................................................................................................. 149

Chapter 6: Period Three – Full Participation .................................................. 152
6.1 Introduction ............................................................................................... 152
6.2 Overview .................................................................................................... 152
6.3 Portfolio Management Framework Review ................................................. 154
6.4 Team Creation ............................................................................................ 158
6.5 Stakeholder Analysis Tool ......................................................................... 162
6.6 Basic Entity Relationship Analysis .............................................................. 169
6.7 Contextual Data Modelling Tool ................................................................. 175
6.8 “Use or Lose” Test .................................................................................... 178
6.9 Quick Architecture Analysis Technique ...................................................... 180
6.10 Summary .................................................................................................. 185

Chapter 7: Business Context & Results ............................................................ 187
7.1 Introduction ................................................................................................ 187
7.2 Business Context of the Third Phase of Study ............................................. 187
7.3 Using the Portfolio Management Framework .............................................. 188
7.4 Portfolio Management Framework for Other Teams .................................. 197
7.5 Summary .................................................................................................... 206

Chapter 8: Conclusion & Further Work ............................................................ 211
8.1 Introduction ................................................................................................ 211
8.2 Restatement of Aim and Objectives ............................................................ 211
8.3 Further Work & Modifications ................................................................... 213
8.4 Researcher Experience of Applying Software Engineering in Industry ........ 214
8.5 Overall Conclusion ..................................................................................... 217

Bibliography ..................................................................................................... 218

Appendix I ......................................................................................................... 231

Appendix I: Estimation and Evaluation Measures and Models ......................... 232
Appendix I.1 Introduction ................................................................................ 232
Appendix I.2 Effort And Cost Estimation Models .............................................. 232
Appendix I.3 Putnam SLIM model for Size ....................................................... 233
Appendix I.4 Financial Evaluation Measures .................................................... 239
Appendix I.5 Summary ..................................................................................... 242

Appendix II ........................................................................................................ 246
<table>
<thead>
<tr>
<th>Appendix II: Quality In-Depth</th>
<th>247</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix II.1 Introduction</td>
<td>247</td>
</tr>
<tr>
<td>Appendix II.2 Approaching Quality from a Holistic Viewpoint</td>
<td>247</td>
</tr>
<tr>
<td>Appendix II.3 Measures of Quality</td>
<td>251</td>
</tr>
<tr>
<td>Appendix II.4 Holistic Quality Models</td>
<td>255</td>
</tr>
<tr>
<td>Appendix II.5 Single Aspect Measures of Quality</td>
<td>257</td>
</tr>
<tr>
<td>Appendix II.6 Summary – Quality Software and Software Quality</td>
<td>257</td>
</tr>
<tr>
<td>Appendix III</td>
<td>263</td>
</tr>
<tr>
<td>Appendix III: Review of IT Successes And Failures</td>
<td>264</td>
</tr>
<tr>
<td>Appendix III.1 Introduction</td>
<td>264</td>
</tr>
<tr>
<td>Appendix III.2 Business Perception of Information Technology</td>
<td>264</td>
</tr>
<tr>
<td>Appendix III.3 Enterprise Resource Planning Software</td>
<td>265</td>
</tr>
<tr>
<td>Appendix III.4 Industry Information System Development</td>
<td>266</td>
</tr>
<tr>
<td>Appendix III.5 Development Successes</td>
<td>267</td>
</tr>
<tr>
<td>Appendix III.6 NASA/University of Maryland/CSC</td>
<td>267</td>
</tr>
<tr>
<td>Appendix III.7 International Business Systems (IBM)</td>
<td>267</td>
</tr>
<tr>
<td>Appendix III.8 AT&amp;T Bell Laboratories</td>
<td>268</td>
</tr>
<tr>
<td>Appendix III.9 Pilkington Glass</td>
<td>269</td>
</tr>
<tr>
<td>Appendix III.10 Anglian Water</td>
<td>270</td>
</tr>
<tr>
<td>Appendix III.11 Sandia National Laboratories Configuration Management System</td>
<td>271</td>
</tr>
<tr>
<td>Appendix III.12 Development Failures</td>
<td>273</td>
</tr>
<tr>
<td>Appendix III.13 Atlanta Olympic Games/IBM</td>
<td>273</td>
</tr>
<tr>
<td>Appendix III.14 Nike</td>
<td>274</td>
</tr>
<tr>
<td>Appendix III.15 Department 56</td>
<td>275</td>
</tr>
<tr>
<td>Appendix III.16 Federal Aviation Authority</td>
<td>276</td>
</tr>
<tr>
<td>Appendix III.17 Internal Revenue Service</td>
<td>277</td>
</tr>
<tr>
<td>Appendix III.18 National Air Traffic Services Centre</td>
<td>278</td>
</tr>
<tr>
<td>Appendix III.19 Industry Experience of Large IT Systems - Summary</td>
<td>279</td>
</tr>
<tr>
<td>Appendix IV</td>
<td>285</td>
</tr>
<tr>
<td>Appendix IV: Software Metrics</td>
<td>286</td>
</tr>
<tr>
<td>Appendix IV.1 Introduction</td>
<td>286</td>
</tr>
<tr>
<td>Appendix IV.2 Types of Complexity Metrics</td>
<td>286</td>
</tr>
<tr>
<td>Appendix IV.3 Use of Regression Analysis of Reported Software Defects</td>
<td>293</td>
</tr>
</tbody>
</table>
Table of Contents

Appendix IV.4 Fitting the Line ................................................. 295
Appendix IV.5 Goodness of Fit .................................................. 295
Appendix IV.7 Summary .............................................................. 298

Appendix V .............................................................................. 303
Appendix VI .............................................................................. 304
Example Questionnaire .............................................................. 304
Record of Contribution ............................................................... 307
Head of Support ....................................................................... 307
Appendix VII ............................................................................ 308
Publication List ........................................................................ 308
Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>SSM Process (adapted from [Vidgen, et al, 1993])</td>
<td>44</td>
</tr>
<tr>
<td>3.1</td>
<td>Adapted from Järvinen's [2001] Taxonomy of Research Methods</td>
<td>60</td>
</tr>
<tr>
<td>3.2</td>
<td>Gantt Chart Representation of Activities for Stage One</td>
<td>74</td>
</tr>
<tr>
<td>3.3</td>
<td>Gantt Chart Representation of Activities for Stage Two</td>
<td>75</td>
</tr>
<tr>
<td>3.4</td>
<td>Gantt Chart Representation of Activities for Stage Three</td>
<td>76</td>
</tr>
<tr>
<td>4.1</td>
<td>Diagrammatic Representation of Stage One Activities</td>
<td>65</td>
</tr>
<tr>
<td>4.2</td>
<td>Example of a Link</td>
<td>67</td>
</tr>
<tr>
<td>4.3</td>
<td>Example of a Link and Précis</td>
<td>67</td>
</tr>
<tr>
<td>4.4</td>
<td>Example of an Emailed Précis</td>
<td>69</td>
</tr>
<tr>
<td>4.5</td>
<td>Cost and Ease of Change Diagram Through Life (from [Fabrycky and Blanchard, 1991])</td>
<td>73</td>
</tr>
<tr>
<td>4.6</td>
<td>Example of Original Feature Analysis Tabular Output</td>
<td>82</td>
</tr>
<tr>
<td>4.7</td>
<td>Example of Original Feature Analysis Graphical Output</td>
<td>82</td>
</tr>
<tr>
<td>5.1</td>
<td>Gantt chart for Period Two</td>
<td>107</td>
</tr>
<tr>
<td>5.2</td>
<td>Option Summary Flowchart - Database does not reflect business practices</td>
<td>110</td>
</tr>
<tr>
<td>5.3</td>
<td>Option Summary Flowchart - Database does not communicate with other databases</td>
<td>112</td>
</tr>
<tr>
<td>5.4</td>
<td>Summary Flowchart - Users do not like using the database</td>
<td>113</td>
</tr>
<tr>
<td>5.5</td>
<td>Feature List Hierarchy Example</td>
<td>115</td>
</tr>
<tr>
<td>5.6</td>
<td>Example of Hierarchical Feature List with Weightings</td>
<td>117</td>
</tr>
<tr>
<td>5.7</td>
<td>Feature Analysis - System View</td>
<td>119</td>
</tr>
<tr>
<td>5.8</td>
<td>Summary Graph of Feature analysis System Comparison</td>
<td>120</td>
</tr>
<tr>
<td>5.9</td>
<td>Feature Analysis, Process View</td>
<td>122</td>
</tr>
<tr>
<td>5.10</td>
<td>Strategic Intent Summary View</td>
<td>123</td>
</tr>
<tr>
<td>5.11</td>
<td>Feature Analysis List - Strategic View</td>
<td>124</td>
</tr>
<tr>
<td>5.12</td>
<td>Hierarchy of Software Defects</td>
<td>135</td>
</tr>
<tr>
<td>5.13</td>
<td>Proportion of Developer Time Taken Per File</td>
<td>138</td>
</tr>
<tr>
<td>5.14</td>
<td>Average Developer Time Spent on Updates for Each File</td>
<td>138</td>
</tr>
<tr>
<td>5.15</td>
<td>Total Update Frequency for All Files</td>
<td>139</td>
</tr>
<tr>
<td>5.16</td>
<td>Cumulative Frequency of Patches</td>
<td>140</td>
</tr>
</tbody>
</table>
Table of Figures

<table>
<thead>
<tr>
<th>Figure 6.1: Overview of the Third Phase of Study</th>
<th>152</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 6.2: Pictorial Representation of the SSM and the Portfolio Management Model</td>
<td>158</td>
</tr>
<tr>
<td>Figure 6.3: Example of a Stakeholder Matrix</td>
<td>163</td>
</tr>
<tr>
<td>Figure 6.4: Example of a Stakeholder Listing</td>
<td>164</td>
</tr>
<tr>
<td>Figure 6.5: Example of a Summary Graph – Internal Stakeholders by Function</td>
<td>166</td>
</tr>
<tr>
<td>Figure 6.6: Stakeholder Summary – Functional Stakeholders</td>
<td>167</td>
</tr>
<tr>
<td>Figure 6.7: Example of Simple Entity Relationship Diagram</td>
<td>171</td>
</tr>
<tr>
<td>Figure 6.8: Example of Contextual Dataflow Diagram</td>
<td>177</td>
</tr>
<tr>
<td>Figure 6.9: Use/Lose Decision Tree</td>
<td>179</td>
</tr>
<tr>
<td>Figure 6.10: Example of the Quick Architecture Analysis Technique</td>
<td>181</td>
</tr>
<tr>
<td>Figure A.1 Cumulative Probability Distribution Graph for Two Values of $\alpha$</td>
<td>252</td>
</tr>
</tbody>
</table>
Table of Figures

Table of Tables

Table 2.1: Tom’s notation .............................................................. 25
Table 2.2: Summary of Lehman’s Eight Laws .......................... 30
Table 3.1: Summary of Research Approaches [Iivari, 1991] .... 58
Table 3.2: Relevant Situations for different Research Strategies [Yin, 1989] .... 60
Table 3.3: Summary Table of Action Research (Denscombe [2002]) ... 69
Table 3.4: Summary of Participant Observation (Denscombe [2002]) ... 71
Table 4.1: Summary of the use of Scare Stories within Rolls-Royce Naval Marine 70
Table 4.2: Lifecycle Costing Summary .................................................. 75
Table 4.3: Summary Table of the Use of Expert Judgement ............. 78
Table 4.4: Original Model Feature Analysis Summary Table ............. 84
Table 4.5: The Strengths and Weaknesses of Legacy Options ......... 99
Table 5.1: Summary of Knowledge Base Application ....................... 114
Table 5.2: Summary of the Use of Modified Feature Analysis – System View .... 128
Table 5.3: Summary of the Use of Modified Feature Analysis – Process View .... 129
Table 5.4: Summary of the Use of Modified Feature Analysis – Strategic Intent View .......................................................... 130
Table 5.5: Summary of the Use of Modified Feature Analysis – Cost View .......... 131
Table 5.6: Summary of the Use of Rich Pictures ............................... 134
Table 5.7: Summary of Experience Gained from Applying Hierarchical Defect Analysis .......................................................... 137
Table 5.8: Experience of Using Graphical Defect Analysis ................. 142
Table 5.9: Summary of Statistical Analysis of Failure Data .......... 143
Table 5.10: Summary of the Lessons Learned from Questionnaire Application .... 147
Table 6.1: Summary of the Stakeholder Management Tool ............ 169
Table 6.2: Basic Entity Relationship Analysis Summary Table ........... 174
Table 6.3: Contextual Data Flow Diagram Summary Table .............. 178
Table 6.4: ‘Use or Lose’ Test Summary Table ................................. 180
Table 6.5: Quick Architecture and Systems Analysis Summary Table .... 184
Table 7.1: PDM Team Tool Use in Stage One ................................. 190
Table 7.2: PDM Team Tool Use in Stage Two ................................. 192
Table 7.3: PDM Team Tool Use in Stage Three ............................................ 195
Table 7.4: Stage One Tool Use ...................................................................... 197
Table 7.5: Summary of Stage Two Tool Use ................................................ 201
Table 7.6: Stage Three Tool Use Summary Table ........................................ 205
Table 7.7: Summary of Researcher’s Experience with the Tools and Methodologies ................................................................. 208
Chapter 1: Introduction

1.1 Introduction

This chapter introduces the research goal of the thesis, followed by a review of Rolls-Royce as a company and a culture. Leading on from the outline of these, a brief examination of the research rationale is put forward. The aim of the thesis, the creation of a tool kit for effective IS portfolio management, is then stated. Objectives are proposed that will lead to the accomplishment this aim, finally rounding off with an outline structure of the thesis itself.

1.2 The Research Goal

The research goal of this thesis was to construct a toolset for IT portfolio management, and test it in an industrial environment. The toolset is to be created in an iterative fashion, allowing business requirements to be fully incorporated and embedded within the toolset. The iteration of tools will increase the ability of the toolset to portray industrial requirements, and increase the usability of the toolset within industry.

1.3 Rolls-Royce – The Company

Rolls-Royce plc is a renowned, large, engineering firm. It is the second largest producer of aero engines in the world and sells products in 34 countries worldwide. The Rolls-Royce brand is a globally recognised brand, standing for excellence in engineering.

The company initially started producing cars under the Rolls-Royce brand at the beginning of the last century. However, the company is now diversified into aero engines, gas turbines, and marine products. The car brand is no longer part of the Rolls-Royce plc product suite.
Due to recent acquisitions, such as Vickers in the 1990’s, Rolls-Royce has become the world’s largest single supplier of marine engineering products and services. Brands include Rolls-Royce, Crossley Pielstick, Bergen, Michel Bearings, Kamewa waterjets, Mermaid propulsors, and Ulstein Design. The Rolls-Royce marine product suite now extends to offer entire boat design, propulsion trains, power plants, and deck machinery.

Rolls-Royce has three main operational areas. The first produces aero engines. This unit has a long history of engine manufacture, producing the famous Merlin engine that powered the Spitfire and the Lancaster bomber, as well as many others. Rolls-Royce aero manufactured the world’s first commercial jet engine, and also the world’s first VTOL engine, the Pegasus, which is in every Harrier plane in the world. More recently, the aero engines have been used in the Eurofighter Typhoon, and the Joint Strike Fighter. The company is famous for the rugged reliability of its engines, especially the RB-211, which has powered many commercial craft. Engines are manufactured for both Airbus and Boeing, and Rolls-Royce also manufactures the engine plant for the Osprey, the world’s first tilt-rotor plane. This unit comprises the largest part of Rolls-Royce, both by turnover and number of employees.

The second area is responsible for the design, manufacture and supply of gas turbines. These are essentially scaled up versions of the aero engines for use in marine and land-based power generation. This is an area where Rolls-Royce has considerable expertise, stemming from the aero engine area. The latest gas turbines can generate up to 50MW of power.

The final area is marine products. This is a relatively new area for Rolls-Royce, and it has only become viable following a series of acquisitions. It is Rolls-Royce’s stated intent to grow this area of their business.

1.4 Rolls-Royce Marine

Rolls-Royce Marine is a business of two halves. The first half provides commercial marine solutions. This covers vessels from small fishing skiffs to huge passenger liners and oil tankers. The second half provides similar services and products to naval
military customers in 55 navies worldwide. The traditional Naval business for Rolls-Royce has been the design, manufacture and support of the Nuclear Steam Raising Plant — the nuclear reactor that powers every nuclear submarine in the UK flotilla. Rolls-Royce was chosen for manufacture because of its history of excellent engineering.

The additional military business has only been recently acquired, through the takeover of Vickers Marine in 1998. Consequently the Nuclear business site of Rolls-Royce Marine has long operated as an essentially autonomous business, having little in common with the other areas of business. It has only been in the last five years that Rolls-Royce has been able to integrate the Nuclear business site into the Rolls-Royce business as a whole. This integration is problematic due to the divergence that has occurred over that last few decades, and the difference in cultures.

1.5 Rolls-Royce Culture

The Rolls-Royce culture is one of perfectionist engineering. The reputation Rolls-Royce has earned over the years has been well deserved. Attention to detail, and design excellence are commonplace. In the Nuclear business site, attention to safety has also become paramount.

The UK is alone amongst the nations with a nuclear flotilla, in never having had a serious accident. The safety protocols that Rolls-Royce have produced, and maintain, have contributed to this in no small part. This is an enviable state of affairs, one which Rolls-Royce is keen to continue.

One other aspect is that of security. Security is paramount within the Nuclear business site, as one would expect, given the nature of the business conducted.

These three aspects, excellence, safety, and security, are central to Rolls-Royce culture at the Naval Nuclear business site. Whatever work is performed, it will have to be culturally acceptable.
1.5.1 Implications on the Research

There are several implications of this culture. The first is that some of what occurs during the research is subject to stringent security controls. Consequently, it cannot be published.

The second is that the peculiar situation of Rolls-Royce in this area, the only supplier of a product for which there is only one customer, creates a unique environment. This has largely insulated the business from the stresses and strains that affect many other commercial enterprises, and substituted a different source of shocks. Because of the close relationship with the customer, the timeframes of business are much longer than might be expected, often extending to decades. Any proposed change to systems must be able to last for a goodly length of time, anywhere from several years to several decades, the business churn being so low.

Another peculiar impact that the product has had on the business is the designation of certain IT systems. Most companies have designated 'business critical' applications — applications that are crucial to running the business, and so downtime must be limited to the absolute minimum possible. The Naval Nuclear business site also maintains 'product critical' applications — applications that are critical to the maintenance of the product. Applications such as these cannot be lost, and data must be kept for long periods of time. This adds to the complexities of managing the IT portfolio — the abundance of legacy data that must, by law, be kept and maintained, is phenomenal. Traditionally, data and the systems that store it are very closely linked. If a system is to be replaced, the requirement to be able to access legacy data means that some form of data migration would be required. Data migration will have an associated cost, thus increasing the cost of any potential migration. There are measures that can be taken to reduce this data load (data migration requirements have to be addressed as part of the IS portfolio management strategy), but the data must be accessible. This quite often means that the path of least resistance lies with maintaining the old system. However, as will be discussed later, maintaining the status quo is not necessarily effective and efficient in the long term.
1.6 Research Rationale

It is one of the fundamental principles of economics that the owners of firms are profit maximising. Profit maximisation entails the owners making decisions with the aim of maximising profit [Coase, 1937]. It can be shown that, in a world of complete certainty and perfect capital markets, there are no consequences of the split of ownership from control vis a vis profit maximisation [Gravelle and Reese, 1996]. However, these circumstances do not exist in the real world. Managers always face uncertainty, and capital markets are not perfect. Therefore there is a consequence to the ownership/control split, and this has been observed.

There have been many attempts to produce alternatives that describe observed actions, within the framework of managerial capitalism, using sales revenue maximisation [Baumol, et al, 1982], growth [Gravelle and Reese, 1996], or the utility gained from ‘perks,’ [Schmalensse & Willig, 1989] as the motives for managerial decisions. All of these models expressed the interests and influence of the shareholders as a single, minimum profit constraint, with variations above the minimum profit being the results of decisions taken by management in line with the maximand. Advances using a game theoretic approach have yielded principal-agent theory using moral hazard and adverse selection. However, none of these models are perfect (see Gravelle and Rees [Gravelle and Reese, 1996] for more details).

The assumption of profit-maximisation is intuitively sensible, and is still used to explain many managerial decisions. As long as the profit-maximisation motive holds, management will always choose to invest in opportunities that will lead to the maximisation of profit. A decision to invest in information technology should therefore come under the same rules governing all business decisions – the IT should enable profit maximisation. Profit maximisation through IT has two possibilities – either the IT will increase efficiencies, reducing costs, or it will increase revenues in total, or both. Maximising profit through either of these methods will require development of an IS portfolio management strategy to develop and implement the capability.
Chapter 1

Introduction

To manage an IT portfolio, a suite of tools is required that:

1. Reduce uncertainty
2. Define the business requirements
3. Define the IT requirements
4. Define the strategic requirements
5. Highlight areas of inefficiency
Chapter 1

1.7 Aim
The aim is to create and evaluate the effectiveness of a tool set for cost effective IS portfolio management.

1.8 Objectives
The aim will be satisfied by the following objectives:
1. To study successes and failures in IT/IS management, identifying the lessons learned, with particular reference to those pertaining to legacy systems.
2. To examine a number of established tools and methods for IS portfolio management at Rolls-Royce Naval Marine and determine if and how these tools can be adapted to better suit the Rolls-Royce environment. The success of these tools will then be evaluated.
3. To develop new tools and methods for use at Rolls-Royce Naval Marine and evaluate their success.
4. To test the general applicability of the tools and methods by applying the techniques to the wider Rolls-Royce company, or at least by gaining company acceptance to use these tools and methods in this wider context.

The first objective looks for lessons learned from other industry experiences. What are common factors in both successes and failures? Replication of the common factors of success, and avoidance of the common factors of failure will increase the probability of success.

The second objective will examine existing tools and evaluate them for use within context. How applicable are they, given certain conditions? Can they and should they be adapted? Will they assist in replicating common success factors and avoiding common failure factors?

The penultimate objective recognises that some of the tools may not be effective in context and cannot be adapted to be so, or that there may not be any tools with the required characteristics. Consequently, new tools may have to be developed, and existing tools may need adapting.
The last objective tests the entire tool set in an industrial environment, critically examining those tools that are used, formulating lessons learned, re-applying the lessons learned, and using the tools in other business units of Rolls-Royce.

1.9 Thesis Outline

The thesis is composed of eight chapters. Chapter two performs a literature review, covering areas of particular importance to the situation faced at Rolls-Royce, but is also equally applicable to other companies facing an ageing portfolio of IT systems. This chapter covers a review of reported portfolio management techniques, the concept, definitions and causes of legacy systems, and finally technical and quality standards that may be applicable.

Chapter three reviews methodological techniques and practices, looking at research approaches and strategies. There is a debate concerning the relative merits of different approaches, before deciding upon a mix of compatible concepts, with the need to be cognisant of the culture within Rolls-Royce – some approaches being more acceptable to Rolls-Royce than others. A grounded theoretical approach to tool and methodology evolution was adopted, twinned with an active participant observation role in the workplace to introduce these tools and methodologies.

The researcher's experience of applying various tools and methodologies is covered in chapters four, five, and six. Each of the three chapters represents a time period of study. The researcher has divided the three year period of the thesis into three distinct sections. The first, covered in chapter four, is the introduction to the workplace. It is the essential background research that is required before the researcher could be accepted as part of the working environment.

Chapter five covers the second of these periods, what could loosely be termed the 'pilot period.' Various early tools and methodologies were tried out by a small number of people in a single department within a single business unit of Rolls-Royce. This presented an opportunity for the researcher to establish a professional credibility to Rolls-Royce as a whole, demonstrating the validity of the concept.
Chapter 1

Chapter six is the last of the 'experience' chapters. This covers the extended and in depth use of the tools and methodologies, showing how they developed and evolved through feedback from use. The tools were tried out across a business unit of Rolls-Royce, with many departments using them.

Chapter seven presents the results. It places the results in a business context, showing the experience of the various teams that used the tools and methodologies.

The final chapter, eight, concludes the thesis, referring back to the alternative portfolio management tools and methodologies discussed in chapter two, and how the tool set presented differs from these. It also offers the advantages of using this tool set. It finishes with a view on further work that could be conducted and discusses the problem of the lack of credibility of software engineering techniques in industry.

1.10 Summary

The research imperative for this thesis stems from the need, by Rolls-Royce, to effectively manage a large portfolio of information systems and technology. Effective and efficient management of the portfolio will have tangible benefits for Rolls-Royce, preferentially in the area of operational cost reduction. An optimisation of IT assets would also be looked for, all in the cause of creating the largest benefit for the company owners.

Information technology is an area traditionally fraught with difficulty, and any tools that reduce the risk and difficulty associated with the effective and efficient management of IT are to be welcomed.

This chapter introduces the thesis, the aim of which is to provide just such a tool set for Rolls-Royce.
References:


Chapter 2: Literature Review

2.1 Introduction

The management of a portfolio of IT systems will inevitably revolve around change. Change seems to be the only constant in the IT environment. With the current rate of change, IT systems rapidly age and become out of date, resulting in this change driver. ‘Legacy systems’ are a term that has been coined to reflect this change, whereby systems become obsolete. Indeed, the term ‘legacy system’ is often used as the reason why businesses change their systems. But what is a legacy system, and why is it considered such a bad thing for companies to possess them? This chapter starts by exploring some suggested causes for legacy systems, and then looking at some of the common definitions of legacy systems.

The chapter then continues by examining some quality standards and measures that would have to be employed when considering developing a replacement for a legacy system, and suggests why developing a ‘quality’ IT system across an enterprise can be so difficult, with the root cause being the difficulty in reconciling different opinions, standards and measures of quality.

The development of any tool set must review what is currently available in order to determine why it does not meet the requirements of the management. The first and second objectives stated in chapter one will be accomplished within this chapter and Appendix Two.

2.2 IT System Portfolio Management

The traditional approach to IT portfolio management in industry is reactive - waiting for problems to present themselves and then dealing with them. The rationale behind this is a combination of extremely rapid pace of technological change (making it difficult to be proactive, when circumstances forcing a reactive change occur so quickly) and a disconnection between ‘the business’ and IT (IT being regard as a universal panacea for all problems). An IT arms race developed between companies, obliging many large companies to hop on multiple bandwagons without giving
thought to the consequences. (The ERP, CRM, and e-anything bandwagons are good examples of this).

The result of two decades of computerisation is that each company will have a variety of systems, languages, protocols, and platforms within itself. Administration of this varied conglomeration is becoming rapidly more complex. A rise in complexity leads to a rise in the operational costs. This is not a cost-effective situation.

Despite this need for change, the published literature on pro-active IT system portfolio management is curiously sparse. Powell [2001] studies time-constrained software development, showing the need for appropriate, detailed, and effective project and programme management. Powell makes particular note of PEL (Process Engineering Language) – a standardised means of described and ordering processes and their descriptors. The study is focussed on manufacturing products, and not specifically on the systems themselves as a business function. The IT function is embedded within the manufacturing function. However, the study does not consider ‘ancillary’ systems, such as ERP, and their impact on the company. The methodology is also fairly complex.

Tom [1990] introduces some models of software accumulation, based upon the premises put forward by Lehman over a number of years (see 2.4.1 for a review). Tom also makes an interesting point that the quality of existing software could be simply measured by the cost of maintaining it. Tom introduces the notations given in Table 2.1.
Using the notation above, Torn proposes a software lifetime of \( k \) years, with software recreation and libraries. Consequently the maintenance, \( m \), is given by:

\[
m = pd \sum_{i=1}^{k} (1 + q)^{i-1} = pd \frac{[1 - q^k - 1]}{q}
\]

Equation 2.1

Using this and associated models, Torn shows that the high resource requirements of software maintenance can be predicted, and shows that quality has an essential role in minimising software maintenance expenditure.

However, Torn makes a number of assumptions:

- The accumulated value of the software \( (V_n) \) should be at least equal to the development cost of the system
- Software complexity is a constant and fully measurable
- Socio-political factors have no influence on software maintenance or development
• Software development occurs on a like for like basis, with very little radical business or technological change, other than that which is predictable on a yearly basis.

Tom’s produces an acceptable model but with many drawbacks, principally due to the assumptions made. Firstly, socio-political factors (soft issues) have an overwhelming impact on software development and maintenance within industry as a whole. In addition there are many cases, (see Appendix Three) where the value of the system to the business is considerably less than the cost of the system itself.

Significantly, Tom makes no references to proactively managing the accumulation, merely noting that it would be possible to optimise the accumulation. The researcher suspects that the optimisation would make similar assumptions to those laid out above, and would thus be of less use then would have been the case. Since 1990, Tom made reference to software accumulation (Eriksson and Tom [1991], Ali, Storey and Tom [1997]), but not from a management standpoint.

Building on the work of Favarro [1996], Erdogmus and Vandergraaf attempt to bring options pricing theory to IT portfolio management. Starting with the traditional Black-Scholes model [Black and Scholes, 1973] and developing them to suit the management of an IS portfolio. This approach has generated interest from both the IT and finance communities (Erdogmus [1999], ibid [2001], Favarro [1996], Kumar, [1996]).

However, the options model employed by likes of Favarro and Erdogmus is known to be flawed (Hull, [1997], Homes and Butterworth [2000a], [2000b]). An implicit assumption of the model is that the system it models does not use any form of future prediction model! Introduction of future prediction leads to instability and mispricing (in the financial world). The researcher assumes that something similar would happen in the IT respect. A game theoretic approach has been suggested to offset this problem in the creation of financial models [Rasmusen, 1995]. However, this approach is highly complex and mathematical and success has not been reported or visible.
In summary, the reported research regarding the management of an IS portfolio is sparse. The researcher suspects that much research has been performed by management consultants on this subject, but has not been reported due to commercial considerations. Of the work that has been reported, there is very little that offers a proactive management approach, one which takes account of socio-political factors. The researcher asserts that a management tool set, aimed to increase information and knowledge available to those who will have to decide portfolio management strategy is not only viable but necessary.

The definition of a tool set has to consider first causes – what is the problem facing the portfolio manager, and why is it a problem. It is easy to articulate the problem at a high level – the problem is one of accumulation; accumulation of systems, languages, technology, and capability. Inevitably, this leads to obsolescence and replacement. This problem is usually summed up by the words “legacy system.”

2.3 Introduction to Legacy System – Causes and Definitions

Legacy systems are a continued presence in the modern business environment [Computer Weekly, 26/07/01]. There are many news stories about how companies have tried to replace legacy systems and failed, with drastic consequences. It is obvious that a legacy system often forms part, if not all, of a large corporate information infrastructure. However, what circumstances create a legacy system? What actually is a legacy information system? Evolution, a process more commonly recognised in biology, could provide an explanation. Codifying the process of evolution, with respect to information systems, has resulted in Lehman’s Laws [Lehman and Belady, 1985]. These laws provide one explanation of the causes of legacy systems.
2.4 System Evolution

**Maintenance, v.t.** To keep in working order

**Evolution, n.** the act of unrolling or unfolding; gradual working out or development

*Chambers Twentieth Century Dictionary, 1961*

A software system is generally produced to a specific set of requirements. However, as Belady and Lehman observed, once installed, the application users invariably find it opportune to use the system differently, or for a different purpose, than originally intended [Belady and Lehman, 1983]. This expansion of use and functionality can be termed evolution. The concept of evolution implies that, given changing environments, unless complex systems evolve, the changing environment will leave them unable to perform any functions, even the ones they were originally capable of performing. Belady and Lehman noted that this applies to large programs as well; “if [they] do not evolve, they will fall into obsolescence and uselessness.” [Belady and Lehman, 1983]

Large systems tend to have long life spans, and therefore must evolve to remain ‘in working order’. Closed feedback loops to provide vital information about the status of the system to the system, necessary for self-regulation. These loops constitute the basis of software maintenance and evolution. According to Weiderman et al. [2000] evolution is “a coarser grained, higher level, structural form of change that makes the software systems qualitatively easier to maintain.”

2.4.1 Lehman’s Laws

Lehman’s Laws, [Belady and Lehman, 1983] [Lehman and Belady, 1985], provide insight into the circumstances that create legacy systems, building upon the concept of evolution. These laws of software evolution have been considerably updated and modified since their inception in the early 1970’s, and have become much more relevant to the practice of software engineering. It is a tautology that any system that works, will be used. It will continue to be used until it no longer works. Therefore legacy systems will always work, after a fashion, otherwise they would not become
Chapter 2 Literature Review

legacy. However, given that code is malleable, and that business requirements change over time, evolution is inevitable.

2.4.2 E and S Type System Classification

Lehman’s Eight Laws of Software Evolution have been gathered from several large projects [Lehman and Belady, 1985] [Woodside, 1980]. To assist the enumeration of these Laws, a simple classification of computer systems has been developed. The E- and S-type classification system allocates a computer system into one of two classes. An S-type system is one that is completely and totally defined by, and is required to be correct with respect to a mathematically defined specification.

An E-type system, on the other hand, resolves to expectations of the system. An E-type system is correct when it satisfies the user expectations. These expectations are from any aspect of the system, such as quality, performance, user-friendliness, or any measure of system functionality or capability. The Laws are principally concerned with E-type systems.
2.4.3 The Eight Laws of System Evolution

Table 2.2 summarises Lehman's Eight Laws (taken from [Lehman and Belady, 1985] and [Lehmann, Perry and Ramil, 1998])

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Law</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Continuing Change</td>
<td>E-type systems must be continually adapted else they become progressively less satisfactory in use</td>
</tr>
<tr>
<td>2</td>
<td>Increasing Complexity</td>
<td>As an E-type system is evolved, its complexity increases unless work is done to maintain or reduce it</td>
</tr>
<tr>
<td>3</td>
<td>Self Regulation</td>
<td>Global E-type system evolution processes are self-regulating</td>
</tr>
<tr>
<td>4</td>
<td>Conservation of Organisational Stability</td>
<td>Unless feedback mechanisms are appropriately adjusted, average effective global activity rate in evolving E-type systems tends to remain constant over product lifetime</td>
</tr>
<tr>
<td>5</td>
<td>Conservation of Familiarity</td>
<td>In general, the incremental growth and long term growth of E-type systems tends to decline</td>
</tr>
<tr>
<td>6</td>
<td>Continuing Growth</td>
<td>The functional capability of E-type systems must be continually increased to maintain user satisfaction over the system lifetime</td>
</tr>
<tr>
<td>7</td>
<td>Declining Quality</td>
<td>Unless rigorously adapted to take into account changes in the operational environment, the quality of E-type systems will appear to be declining</td>
</tr>
<tr>
<td>8</td>
<td>Feedback System</td>
<td>E-type evolution processes are multi-level, multi-loop, multi-agent feedback systems</td>
</tr>
</tbody>
</table>

Lehman’s Laws indicate how legacy systems occur. Any system, once implemented, will be used if it works. By Lehman’s first law, the system will always be subject to change. Given that the real world requirements are unbounded, any software implementation will, by definition, be incomplete. It will never, and can never, satisfy all requirements at any given time.

The second law illustrates the danger of an evolving system. It can become more complex. The imposition of change upon change creates additional complexity. The more elements there are, the greater the number of potential connections, and the greater the effect of any change. The number of potential connections is equal to \( \frac{n(n-1)}{2} \) components. Thus, the long-term trend for complexity tends towards an inverse square relationship [Turski, 1996]. The more complex any given application
becomes, the longer it takes to modify, and consequently, the cost of modification increases.

The sixth law is very similar to the first law, in that it represents change. However, it represents change due to changing expectations, leading to requirements change, not the requirements gap, as identified in the first law. This law, more than the first, models the effects of evolution on a system.

The seventh law, that of quality, is the final descriptor of a legacy system. The quality of the system degrades as change is implemented on top of change. Quality decreases increasing maintenance costs, as the decrease in quality can be marked by an increase in complexity.

Once the effects of these laws are combined, it is easy to see what creates any given legacy system. The first and sixth laws dictate that a system must change. The second law states that these changes can increase complexity, which makes the system more expensive to maintain. The seventh law shows that the effects of these changes (as triggered by the first and sixth laws) will cause a decrease in the quality of the system. The end result is an expensive, low quality information system – a true legacy system.

Belady and Lehman's definition is a derived definition from a sequence of processes governed by Laws. But this definition is not a commonly applied definition.
Chapter 2 Literature Review

2.5 Legacy System Definitions

A legacy system is the result of processes described by Lehman’s Laws. But the definition of an E-type system, which is the system to which all of the Laws apply, allows a multiplicity of definitions of what actually is a legacy information system (LIS). A good definition will be one that should take into account a full expression of an E-type system – functionality, the maintainability, and the evolution of the system. It is recognised that, as shown by Lehman, systems’ functionality changes over time. However, it is assumed that systems are not infinitely scalable, expandable and extensible. Maintainability represents the ‘in-service’ aspect of the system. It portrays the effort required to maintain the system in a state in which the functionality is adequate for current needs. Evolution is the extensibility of the system – how far can the system be expanded and extended beyond the design intent?

2.5.1 The Literal Interpretation

Legacy n. 1. A gift left in a will 2. Something handed down by a predecessor


In the literal sense, a legacy information system could be defined as a system that is handed down by a predecessor. At first glance this definition has merit – most of what are called legacy information systems are old systems, and therefore a predecessor generally made the implementation decision. It suggests every system has a potential to become a LIS once the decision to implement has been made. This is also an intuitive truism. However, this definition does not take into account any of the aspects described above as necessary to a good definition of a legacy system. It does not allude to any aspect of functionality or maintainability, and certainly does not account for the evolutionary part of a full definition.

The literal interpretation is further revealed to be fallacious when various scenarios are considered. It suggests that the easiest way to avoid a system from becoming a LIS is to keep the same decision-maker in position. As long as the person who made the decision to implement the LIS is present, the system cannot be a LIS, as it will not have been handed down! It also implies that a LIS can be made ‘non-legacy’ by re-
employing the implementer!

The literal interpretation of legacy system is not very common, and not very useful. The logical flaw in the definition, as pointed out above, renders the definition invalid. However, the researcher has seen this definition employed within certain areas of business, despite its obvious flaws.

2.5.2 Old System Definition

Possibly the most commonly used definition of a legacy system is one that is merely old. It is slightly different from the literal interpretation mentioned above, but it also uses time as the only measure for determining a legacy system. A legacy system under this definition is one that was written greater than $x$ years ago. The exact size of $x$ is not really defined, as there are far too many variables to consider, not least of which is the product lifecycle. Fast-moving businesses, such as Amazon.com and Cisco, have a very rapid lifecycle, leading to obsolescence of information systems within months to years. In other areas, such as engineering, the rate of change of information needs is low, and therefore IT system life spans can be measured in decades.

The old system definition is a very useful one, although, like the literal definition, it is a tautology. It references the functionality, maintainability and evolutionary aspects of the Belady-Lehman definition indirectly – an older system is more likely to not have the functionality, maintainability, and evolutionary problems ascribed by the Belady-Lehman definition. There is the perception that legacy systems, as defined this way, are a problem. A system could be labelled as legacy, and thus targeted for replacement, even though it is functional and maintainable.

2.5.3 Unknown System Definition

Before the existence of standard, structured development methodologies such as SSADM, IT systems were created in a very haphazard way, commonly by one person, or a small group of people. The unstructured design and implementation created systems that operated in a ‘black-box’ fashion. Managers of such systems did not know how these systems operated, only that they did. Knowledge of the internal
workings of the systems was retained in the heads of the few computer developers.

Maintaining such as system becomes problematic. Scheduling updates are difficult, as it is hard to predict the behaviour of the system. The system is critically dependent upon the original creators for continuing stability.

This definition introduces a human element into the assessment of whether a system can be considered legacy. A legacy system of this type would be immediately identifiable, as it would depend upon a very small group of people in order to continue working. Systems of this type are generally quite old, which suggests that this definition has many elements in common with both the literal interpretation and the old system interpretation.

2.5.4 Maintenance Problem Definition

Bisbal et al [1999] defines a legacy system as "any information system that significantly resists modification and evolution." This interpretation has many of the features missing in the literal interpretation. It explicitly mentions modification and evolution. In other words, a LIS is impossible to expand to cope with future demands.

But, again, there are holes in the argument. By this interpretation, any COTS product is automatically a LIS, as it will resist modification by design. For example, Microsoft's Windows operating system significantly resists modification and evolution outside of pre-determined parameters. Unless you are Microsoft, it is very hard to extensively modify a Windows implementation. Is Windows a legacy system? Or any COTS product?

Another shortfall in this argument is that it does not make any reference to functionality. A system may be perfectly fit for purpose, and thus there is no need to replace it in the immediate future, as long as needs remain constant. The decision to address the issue can be deferred. But this functional system can still resist modification and evolution. Obviously this definition is better than the literal interpretation, but there are still significant flaws. Again, these flaws do not appear to restrict the use of this definition.
2.5.5 Pervasive and Uncontrollable System Definition

Bennett [Bennett, 1999] presents an informal definition - "large software systems that we don’t know how to cope with but are vital to our organisation.” This is a very broad definition, and can include all aspects of what makes a good definition as defined above.

However, this interpretation can be misleading. For example, suppose that the detailed operating knowledge was contained in a small group, which then disperses. The remaining IT staff are not available to maintain the system. Therefore this system is now classified as a LIS, as it cannot be coped with, given the present circumstances. If IT professionals are then hired, this skill shortfall is removed, and the organisation is now able to cope with the system. Therefore it can no longer be classified as a LIS. Another scenario could be a SAP installation. Many companies have exceedingly complex SAP installations. So complex, in fact, that SAP has to maintain a consultancy in order to maintain these systems’ functionality. Under this definition, these systems are also LISs, even though they may be perfectly maintainable and supportable in their current form. Again, this definition seems to fit some circumstances, but cannot be applicable to all ‘legacy’ systems.

2.5.6 Obsolete Language/System Definition

Somewhat similar to the unknown system definition, is the definition of a legacy system as a system that employs an obsolete language. It is then a matter of defining what an obsolete language is. One definition could be that it requires maintaining a specialist team to administer the information system. For example, the programming 'ethos' of today is different to that of the 1970's. During the 1970's and 1980's, memory and processor time were intensely valuable, and programs were tailored to make the best possible use of these rare commodities. This quite often came at the expense of legibility. Modern programming de-emphasises economy of programming in order to emphasise legibility and maintainability.

Many programs were also designed in current languages that have since been superceded. There exist many thousands of programs written in COBOL. COBOL
experience is vanishing, as new programs are generally not written in COBOL. Other such languages include FORTRAN and ADA.

2.5.7 Non-Value System Definition

Weiderman, et al. [Weiderman, et al, 2000] suggest that all IT systems should be looked at in terms of assets and liabilities. “Software is being viewed as an asset that grows over time rather than a liability whose value depreciates over time.” The definition then goes further to say that IT assets that require increasing amount of assets to maintain, without a concommitant increase in asset value, have become stale, and the value of the system decreases. “Eventually, there may be more cost associated with their continued maintenance than benefit from their continued use.” King, [2002] also supports this definition, describing legacy systems as “non-leverageable” and “ear-marked for retirement.”

How can the value of an IS be measured, in order to assess its value to the company? Hard benefits, such as the ability to lay off staff due to automation, are easy to measure. But soft benefits are harder to quantify. What is the value of a better user interface? Is there value in being able to continue business? Or being able to pursue new avenues of business? A system could support current business requirements perfectly capably. But it may not have the facility to expand. Therefore the system is always returning value in the shape of current business, but is also losing value, in that it represents a barrier to further business growth.

What if a new information system supports future contracts, but makes little difference on the present method of business? The IT system does not produce any concrete savings, but does allow the possibility of future income. It does not, and cannot, guarantee this future income. Does this constitute returning value? Does this also mean that a LIS can be made not legacy by finding an alternative use for it that does add value to the company?

Despite these shortcomings, the definition of a legacy system in terms of its value to the company is appealing. It is something that can lead to a metric, if one ignores several of the ‘softer’ elements as mentioned above. It is also one that is intuitively
understandable to business decision-makers, as it will have direct relevance on their ability to deliver to the bottom line. It also has the advantage in that it implies that systems can be re-valued by evolution, both negatively and positively, thus allowing a company to increase the returned value of a system to the enterprise. It is for these reasons that the researcher prefers this definition to all of the others. It also encapsulates much of what the other definitions cover. For example, a 'legacy system' could require a company to maintain an entire staff with a unique skill set, and maintain antiquated servers for the simple reason that the system cannot run on the newer servers. The system would be a legacy system under the obsolete language, aged system, literal and maintenance problem definitions. However, if the system returns a great deal of value to the enterprise, these costs are acceptable, and thus this system would not be considered legacy under the preferred definition.

2.6 Maintenance Aspect of Legacy Systems

Various definitions of legacy systems have been mentioned above. With these definitions, it is possible to see that there would be occasions when legacy systems, in the various definitions, would be an unfortunate occurrence for a company. However, legacy systems are commonly considered to be a colossal problem for businesses, so much so that they are willing to invest many millions annually in removing them. Why is this?

One possible reason is maintenance costs. The application goalposts are continually moving, and to maintain the application for use requires the system to evolve. However, Weiderman et al, [2000] defines software maintenance as “a fine-grained, short-term activity focused on (perhaps a large number of) localized changes.” Maintenance constitutes a large part of the total cost of an IT system [Bradley and Dawson, 1995] [Bradley and Dawson, 1999]. Maintenance could also be considered evolution under the Belady and Lehman concept of information system evolution. Sneed [1999], defines maintenance tasks as either:

- Corrective - the analysis and correction of errors; debugging
- Adaptive - the changing of data formats and algorithms
- Perfective - either the optimisation or the enhancement of existing software
As Sneed’s definitions show, it is difficult to find the dividing line between maintenance and evolution – even corrective maintenance is evolutionary, in that it fixes what is not right. Evolution in IT systems would seem to be more Lamarckian than Darwinian.

The costs of maintaining/evolving a legacy system, by Lehman’s Laws and encapsulated in the non-value system definition, are in excess of those that would be incurred with a non-legacy system. This could explain why legacy systems are considered to be an unacceptable part of the business infrastructure. Systems that cost an excess over the opportunity cost will not be compatible with the profit maximisation motive, as they will directly affect the ability of the manager to deliver to the bottom line. The non-value system definition allows flexibility in this. The costs to maintain a system can rise until they are greater than the value the system returns to the business. Once this occurs, the system is termed legacy, and must be removed. The removal of non-value systems is therefore sound business sense.

However, legacy systems can represent a substantial body of business knowledge, such as validation rules, data structures and so on. Replacement by exact replication would be highly unusual, not to mention pointless, and therefore it would be expected that some of this business knowledge might be lost to the company following a replacement. This knowledge can quite often take the form of age-old assumptions that everyone takes for granted and forget to replicate in the newer system. It could be argued that business knowledge represents a source of value, and thus could be measured. However, as was pointed out earlier, some aspects of value are more difficult to cost than others. Opportunity cost of lost knowledge is a very hard concept to capitalise.

To offset the loss of a functional application and the body of business knowledge, it obliges the strategy developer to look very carefully at which definition of legacy system is being used (i.e. why it is considered to be a legacy system), and what value can be placed on the business knowledge within the legacy system. The options available for consideration within the strategy will be different depending on the definition of legacy system used. For example, if the old system definition is used,
one reasonable option would allow for the system to remain in place. If the system is obsolete in an unusual language, leaving the system in place would not be a sensible (cost-effective) option. Similarly, the amount of maintenance costs can also affect the choice of options that should be considered.

2.7 Quality Standards and Measures within the Business

Managing a portfolio of IT systems will inevitably include the development of new systems and the retirement of older ones. Several of the definitions of legacy systems mentioned above capture the lack of design methodologies and difficulty of maintaining through life these systems without adequate documentation. It is now accepted that one of the aims of any new software engineering project should be to produce a ‘quality’ system or process. Most large companies subscribe to some sort of quality paradigm, relevant ones being described below. However, ‘quality’ seems to be a very ill defined term, even within a single company. A mechanical engineer will have a different perception of quality to that espoused by a librarian. The word ‘quality’ has many different meanings to many different people. Some of this confusion no doubt extends from the different areas in which software engineering is involved. Coders and project managers will view a ‘quality’ software application differently.

2.7.1 Quality Ad Absurdum

“One of the problem areas that business in the New New Economy will have to deal with is the overabundance of Quality, Quality Training, Total Quality Management, and things of that ilk. If you avoid Quality, you do not have to spend precious time and resources managing it. [McEachern and O’Brien, 2002]”

There seems to be an almost overwhelming abundance of quality standards. As the quote above illustrates, there is also a high level of cynicism and scepticism surrounding them. It is easy to see at least one of the reasons for this attitude. To take a hypothetical example, a business wishes to create a software application. How many standards are associated with producing quality? Reconciling these across the
business to produce a commonly acceptable ‘quality’ system will be shown to be highly problematic.

These are some of the main standards governing software quality. The IEEE 830 covers Software Requirements Specifications, as does IEEE 1233 (A Guide to 830, and a standard for developing procedures for collecting requirements). For military organisations, DEF STAN 00-55 Part 1/1, The Procurement of Safety Critical Software in Defence Equipment – Requirements is also applicable. The infamous ISO 9001-2000 is possibly the best known requirements standard. IEEE 828 and IEEE 1042 both cover Software Configuration Management for Quality. ISO/IEC 14143-1 and Mark II use function points and Symons Mark II Function points respectively to attain quality [Symons, 1998]. ISO 9000-3 is the guide document for the application of ISO 9001 to the development, supply, and maintenance of software.

Software maintenance is further covered in ISO/IEC 14764 [ISO, 1998], and to some extent in ISO/IEC 12207, IEEE 1540, IEEE 828 and IEEE 1024. It is also the subject of IEEE 1219. Military organisations may also include IEC 60880, DEF STAN 00-55 Part 2, and DEF-STAN 00-56 Part 2.


Given that software development is a project, there are also some software project management standards that can apply [12207.com, 2002]. IEEE 1028 covers software reviews and audits, with IEEE 1044 a standard for classifying anomalies. IEEE 1045
covers software productivity metrics. IEEE 1058.1 is for software management plans. ISO 15507 also has project management aspects, and there is ISO-IEC TR 16326 that acts as a guide for this project management aspect. IEE 730 and 730.1 cover quality assurance. In specific cases, ASME NQA-1 is also applicable.

While it is noted that ISO standards can supersede IEEE ones, they do not supplant them, the standards do not approach the area in exactly the same way, and do not provide the solution in an identical manner. This list only covers internationally used standards. There are many other organisations, such BSI, ANSI, EIA, ESA, and the FDA to name a few, that have produced standards governing software quality [12207.com, 2002].

ISO standards have incorporated some of the quality models mentioned above, notably McCall's model [McCall, 1977]. However, there are those models and methods that have not been incorporated, such as PRINCE. This additional layer of standards can be self-defeating. How is a company supposed to be assured of its standards, when it is difficult to know which standard to work to?

The number of standards further enlarges when IT-specific standards are considered. Fortunately, there are several de facto standards, such as TCP/IP, HTTP and SOAP, that have industry backing. But, as has been shown above, there are areas where there are competing standards, such as CORBA, DCOM, and so on. There are also those that have polarised the industry, such as MP3/ATRAC3, DHTML/CSS. This 'superabundance' of quality standards could well be one reason why they are not adhered to – it is very difficult to follow all of the quality standards all of the time.

To produce a quality system, a company must agree a definition of quality that will be used. To a project manager, quality quite often boils down to giving the customers what they want [Gitlow et al, 1989] [Turner, 1993]. An adequate and acceptable definition of what quality is, and what and how to measure it is vital for the production of a quality IT system, as required by most companies' quality standards. Quality is required in the software, as the end result has to be the most effective software, and software cannot be most effective with a large number of flaws. Covered below are two relevant quality approaches, SSADM and SSM. Then a
number of measures are discussed that are most relevant to this thesis. Appendix Two covers the topic of quality standards, approaches and methods in more depth.

2.7.2 Structured Systems Analysis and Design Methodology (SSADM)

SSADM has been developed in the UK. Its origins lie with Learmonth and Burchett Management Systems, which developed a methodology for collecting and refining software application requirements, influenced by the works of Yourdon, Merise, and De Marco [Goodwin, 2001]. The first use of the methodology occurred in 1981, with several projects having a quick uptake. The UK Government declared it a mandatory methodology in 1983 [Goodwin, 2001] [Downs, et al, 1992].

SSADM was promoted as an ‘open standard,’ as it was intended to become, like PRINCE, a de facto standard for requirements capture. The project management aspect of SSADM is sparse, as it is intended that the PRINCE methodology is used. Similarly, risk analysis and management is the subject of another methodology – CCTA’s Risk Analysis and Management Method.

SSADM has five core modules:
1. Feasibility Study
2. Requirements Analysis
3. Requirements Specification
4. Logical System Specification
5. Physical Design

The principle is to derive the software requirements and phrase them in such a way as to guide the development process from conception to production. Central to these modules are the graphical aspects of SSADM – the data flow models and logical data model. The logical data model is primarily determined through entity relationship modelling.

The first version of SSADM catered for only the conventional, waterfall development lifecycle, but this has been extended and adapted to cope with different lifecycles, such as rapid application and development.
2.8 Soft Systems Methodologies

Structured, formal analysis techniques, such as SSADM (see [Downs, et al, 1992]), have been found to be highly beneficial in the development of IT systems. However, their use in the early analysis and strategy formulation stages, is dubious as formal methodologies do not explicitly include in the analysis factors that are outside of the IT systems, such as the 'people factors,' politics, and business strategy. These soft factors are considered critical to IT project success. But these factors are not catered for in many of the structured analysis techniques.

To address the deficiencies, soft systems methodologies have been developed. These address not only the IT system, but also the context in which the IT system will be used.

There are several soft systems methodologies. A few of them are the ETHICS methodology by Mumford [Mumford, 1999], Eason's User-Centred Design [Eason, 1988], and Checkland's Soft Systems Methodology [Checkland, 1972] [Checkland, 1981] [Checkland, 1999]. SSM has been previously used at Rolls-Royce, and so it was this tool that was favoured above the others [Ward, 2000].

Checkland's version of SSM was developed during the 1970's [Checkland, 1972] to address the shortcomings of the more formal engineering-based development methodologies. Checkland's original premise was that systems engineering is not entirely applicable to the 'human activity systems,' with lower-level, personal goals often masking the higher-level, organisation goals. Checkland found that IT solutions at the time seemed not to be addressing the real cause of the problems they were experiencing.

As illustrated in Figure 2.1, SSM is a multi-stage process. The first stage is expression of the problem. This involves the analysis of the problem space, identifying stakeholders, and understanding the inter-relationships that exist, and what has caused the problem to be addressed.
The second stage is formulating some relevant models. This is the negotiation phase, where the stakeholders' viewpoints are compared and contrasted, and some common ground is achieved. This stage will develop the conceptual model for the improved system.

Determining desirable and feasible changes constitutes the third stage. This is essentially a feasibility study to determine what can be done to meet the conceptual model. Various options are considered at this point.
The last stage is actioning the preferred option from the previous stage. This is the final stage of implementation, and can be accomplished by using one or more of the more formal methodologies described below.

This technique is ideal for defining problems within an industrial environment, where there are many factors that influence project success or failure, and not all of them are closely related to the IT function.

2.9 Measuring Quality

Quality processes ought to drive out quality outputs, or so the total quality management paradigm states [Oakland, 1995]. But just as quality is hard to define, so is a quality process. Having a metric for quality might make matters easier for a business, as it can then measure to ascertain whether quality is being attained. Sneed [1999] defines quality as “...a relative value that is meaningful only when compared to postulated values that are defined by the user or by standards organizations.” Several people, McCall [McCall et al, 1977] and Boehm [Boehm et al, 1997, Boehm et al, 1978] in particular, have proposed holistic quality models incorporating a wide array of measures, in order to define a quality system. However, these holistic models, as described in the Appendix Two, often require substantial infrastructure in order to capture and analyse the data gathered. Consequently, many companies look for easier alternatives, such as single measure quality, as opposed to process-driven quality.

2.9.1 Defect Density

This measure is commonly used to proxy for software quality. As part of this measure, two defect types are identified. Firstly, there are known defects. These are defects that have been discovered through testing, inspection and user experience. The other type of defect, latent defects, are those that are present in the system, but have not yet been detected. Defect density is then defined as:

\[
\text{defect density} = \frac{\text{number of known defects}}{\text{product size}}
\]

Product size is normally measured in terms of lines of code or function points.
There are several problems with using this fairly crude measure of quality. Firstly, it treats all faults as having equal severity. This is patently not the case. Adams [1984] conducted an analysis of data on nine software products, and related detected faults to their manifestation as observed failures. One of his findings was that about one third of all detected failures lead to the 'minimum likelihood' types of failures – those that occur on average once every 5,000 years of run-time. He also found that a comparatively small proportion of defects accounted for a large proportion of the common failures – 2% of defects would cause failure once in every five years of run-time. The Adams data shows that it is perfectly possible to have an application with a large number of defects, but with a very low failure rate. It begs the question as to whether the application is of high quality in that it does not fail frequently, or low quality in that it contains many defects.

Secondly, the defect density metric is ill defined. The terminology has differed in many studies, and has been used interchangeably with fault rate, fault density, and defect rate [Fenton and Pfleeger, 1997]. The rules as to which faults are included, at what point in the lifecycle are flexible. This means that cross-product quality evaluations are complicated.

Using the defect density to measure quality can also be misleading. Under this philosophy, a zero-defect product should be of the highest quality. Cox [1991] describes some systems at Hewlett-Packard that were described as zero-defect, until it was discovered that these systems had not been used, and therefore no defects had been discovered.

2.9.2 Reliability & Maintainability

Quality can also have connotations of reliability and maintainability – a “quality system” can be taken to mean “one that doesn’t frequently go wrong” or “has low maintenance costs [Oakland, 1995].” Downtime in hours is suggested as the easiest data to collect.
However, management of an IT portfolio will inevitably have a hardware angle as well as the software angle that has been concentrated on so far. Measurement of the quality of the hardware will also drive decisions, and reliability and maintainability of the hardware is an important part of the manager’s role. Consequently, and reliability measure is included here for hardware, namely Weibull analysis.

Weibull analysis is a statistical technique widely used in reliability engineering [O’Connor, 1991]. Its use is primarily due to the flexibility of the Weibull distribution in describing a number of failure patterns. It is of particular relevance when there are several failure modes apparent, as each failure mode is likely to have different shape and scale parameters. It also has the benefit of being able to cope with extremely small data sets. The associated British Standard on Reliability [BS IEC 61649:1997] only requires a minimum of 10 data points. Weibull analysis is described more fully in the Quality appendix, Appendix Two.

2.9.3 Quality Aspects in Software Engineering

Quality is a perceived imperative in software. The recent emphasis on extracting more benefit from IT has only increased the requirement for quality in IT system development and maintenance. However, it is apparent that quality is lacking in software. There are plenty of indications that IT systems have an ‘internal’ quality—the code is effective and efficient; defects are minimised. There are many frameworks and standards that apply that assist in achieving this kind of software quality.

The ‘external’ quality—the system is effective and efficient, business needs are met, and business problems are resolved, is obviously lacking. There are many examples of projects that do not meet identified business problems, or that interfere with the smooth running of the business because of the way in which the systems have been developed [Cushing, 2002].

This internal/external distinction is also picked up by Vidgen, et al [1993], when suggesting use of SSM in deriving quality in IT systems. Their conclusion is that SSM’s socio-organisational approach is more conducive to the development of quality IT systems, as the definition of quality as it pertains to IT is very contextual, and not
totally dependent upon the code of the software – ‘internal’ quality as referred to above.

2.9.4 Quality – Summary

Quality is required in industrial systems. Most companies have signed up to some sort of formal quality management process, and are committed to producing quality outputs. This chapter has reviewed some of the methodologies and measures for achieving software quality. However, it has also discussed how accurate the measures of ‘quality’ are, and whether they are truly representative of software quality. The conclusion is that quality is almost inexpressible, but that ‘you’ll know it when you see it.’ It is suggested that the quality standards do not fully embody ‘quality’ as commonly perceived. It is not suggested that the quality standards are wrong, just that they are not right. Empirical evaluation of the quality standards, and whether they are inherently flawed by weaknesses of their theoretical justification, is outside the scope of this project. What is certain is that quality can be designed, but it is not inherent from merely subscribing to some quality standards. Reconciling quality approaches, standards, measures, and opinions across the enterprise could well be an impossible task.

Satisfying all possible quality standards has been shown to be impossible. It is difficult to even enumerate applicable standards. In terms of this thesis, two quality standards and measures are proposed. The first measure/standard for the solution and the second for the toolset. Rolls-Royce is ISO-9001:2000 certified. Any solution must be developed and rolled out in line with defined processes, in order to remain ISO-compliant. If relevant processes are not available, processes must be developed, again in line with ISO recommendations. The measurement of this quality is a Rolls-Royce requirement.

The second measure of quality will be a practical one and measure the quality of the toolset itself – that of user satisfaction and ‘fitness of purpose.’ The solution will be developed in close association with the users, in an iterative manner, reflecting their needs. The tools that are found to be the most useful in helping to define the implement the solution will, by this definition, be those of the highest quality.
2.10 Review of the Aim

In light of the literature review the aim and objectives can be assessed. The immediate and apparent contribution lies with the entire concept of IT portfolio management. It is apparent from the literature that there is very little of substance published in the public domain. Much of what is there has been shown to be of little use or impact in the management of an IT portfolio in an industrial environment. Proactive management of the IT portfolio is necessary to ensure profit maximisation. Gaining senior management support for proactive portfolio management is vital, and it is the researcher’s opinion that there are no toolsets in the public domain that enable this, and thus the thesis will make a valid contribution to the knowledge in the public domain.

2.11 Summary

This chapter has covered many topics that are important in portfolio management. In terms of the aim of the thesis (creation and evaluation of an IT portfolio management toolset), it is necessary to explore what is currently available in IT portfolio management. This chapter has explored several possibilities, drawing not only from the IS/IT literature, but also from the bodies of financial and mathematical literature. It was shown that the literature is scant on actual IT portfolio management and the tools that were in the public domain are not really suitable as management aids.

Any large company will have an array of systems, and many companies mention that they have a perennial ‘legacy system problem.’ The chapter explores what creates aging systems, and why they can be problematic. It provides several different definitions of the term ‘legacy system,’ before choosing a value proposition definition as the most appropriate in a business setting.

The chapter then explores quality standards and measures, as these will be of high importance in the development of any option regarding IT portfolio management strategy. It shows the bewildering array of quality standards that can be applicable and also demonstrates the difficulty in measuring quality in IT systems.
Chapter 2

The chapter closes with a review of the aim, and a discussion of the contribution this thesis can make to the knowledge in the public domain. The next chapter sets out the research methodology that will be used.
References:

Journal of Research and Development, 28(1), pp. 2-14
optimization algorithms to practical problems", Journal of Optimization Theory and 
Applications 95, No. 3, 545-563.
pp 19-23
Issues and Directions”, IEEE Software, September/October, pp 103-111
Journal of Political Economy, vol. 81, pp. 637-659
Boehm, B.W., Abts, C., Clark, B., Devani-Chulani, S. [1997] COCOMO II Model 
Definition Manual, University of Southern California
TRW Series of Software Technology, Amsterdam, North Holland
Loughborough University
Applications”, in 4th Software Quality Conference, University of Abertay, July 
BS IEC 61649:1997
solving,” Journal of Systems Engineering, Vol. 3, No. 2
Checkland, P. and Scholes, J. [1999] Soft Systems Methodology in Action, Wiley and 
Sons, Chichester
18
Cushing, K. [2002] “Stem the flood of IT waste,” *Computer Weekly*, 18/07/02, pg.22

King, J. [2002] “Make IT bigger, better, faster,” *Computerworld*, 01/01/02


Chapter 3: Methodological Review

3.1 Introduction

The third chapter is, in many respects, a continuation of the literature review. It seeks to review and discuss the many different approaches that are available to a researcher, and pick the most appropriate one for the situation facing the researcher at Rolls-Royce. Although not directly fulfilling either the aims or the objectives, the research approach will have a marked impact on the way in which the aims and objectives are fulfilled.

3.2 Research Philosophy

A research philosophy will determine the way in which data are collected, collated, and analysed. The term epistemology (what is known to be true) as opposed to doxology (what is believed to be true) encompasses the various philosophies of research approach. Given this, the purpose of science is the process of transforming things believed into things known: doxa to episteme. Two major research philosophies have been identified in the tradition of science, namely positivist (sometimes called scientific) and anti-positivist (also known as interpretivist)[Galliers, 1992].

3.2.1 The Positivist Approach

Positivists believe 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 timeless. This approach is has been espoused by many philosophers and scientists, from Plato and Aristotle, through Newton and Bacon, to Popper, Russell, and Planck. All of these philosophers and scientists have made an explicit assumption that the observations are measurable, repeatable, and that the correct analysis will lead to the 'correct' answer, which will remain correct [Popper, 1959]. It is for this reason that the pure positivist approach has found favour in the 'hard' sciences, such as mathematics, physics, and engineering.
There has, however, been much debate on the issue of whether or not this positivist paradigm is entirely suitable for social sciences [Hirschheim, 1985], with many authors calling for a more pluralistic attitude towards Information Systems research methodologies [Kuhn, 1960], [Bjørn-Andersen, 1985], [Remenyi and Williams, 1996]. Indeed, some of the difficulties experienced in IS research, such as the apparent inconsistency of results, may be attributed to the inappropriateness of the positivist paradigm for the domain. Likewise, some variables or constituent parts of reality might have been previously thought unmeasurable under the positivist paradigm, and hence went un-researched [Galliers, 1992].

### 3.2.2 The Anti-Positivists and Interpretivists

As mentioned above, the pure positivist approach has come under fire from several angles. Philosophically, Burrell and Morgan [1979] refer to a group of thinkers that refute some or all of the basic tenets of positivist thought, and rather unhelpfully refer to this school of thought as ‘anti-positivism.’ Anti-positivism seems to have its roots in the Kantian hypothesis that it is impossible to know anything *a priori* about the world as it is not independent of our cognitive apparatus [Kant, 1781]. The Hegelian dialectic [Hegel, 1821] takes this concept even further, as he did not believe that we could know things as they are themselves, and not, as Kant argues, as they appear to us. Hegel created the concept of noumena and phenomena. The phenomena are perceptions of events; noumena are the events themselves. Any recording of an event is phenomenological in nature, and inevitably biased by the very existence of the observer. It is from this that the alternative name for the approach, ‘interpretivist,’ takes form.

Scientifically, the application of the positivist approach to the realm of quantum physics has found that the pure positivist approach breaks down. “I am unsure whether some of these sub-atomic particles exist until we look for them” [Feynman, 1992]. This recognises the essence of both the Kantian and Hegelian positions, stating that there can be many different perceptions of reality, and none of them may actually describe reality *in extremis*. 
3.3 Choice of Research Approach

De Vreede [1995] observes that, in both organisation science and information systems research, interpretive research used to be the norm, at least until the late 1970s. Since that time, however, the positivist tradition has taken a firm hold, Orlikowski and Baroudi [1991] noting that 96.8% of research in the leading US IS journals conforms to this theory. It has often been observed [Benbasat et al, 1987] very accurately that no single research methodology is intrinsically better than any other methodology, many authors calling for a combination of research methods in order to improve the quality of research [Kaplan and Duchon, 1988]. Equally, some institutions have tended to adopt a certain "house style" methodology [Galliers, 1992]; this seems to be almost in defiance of the fact that, given the richness and complexity of the real world, a methodology best suited to the problem under consideration, as well as the objectives of the researcher, should be chosen [Benbasat, 1984], [Pervan, 1994].

It is apparent that a positivist approach in the context of the study area, the management of a portfolio of IT systems, would seem out of place. Every business will face a different environment and drivers, and consequently the approach to the management of an IT portfolio must be different. The researcher has therefore chosen the interpretivist philosophy, which seems more appropriate to purpose. It should be noted at this point that the management of this portfolio is expected to involve the researcher at many levels, and thus the researcher would become part of the system, a position that effectively excludes the positivist school.
3.4 Research Approach

There are many different combinations of research approaches that could have been adopted and used as a framework to undertake the planned research experiments. The researcher has considered three broad styles of research approach. Iivari [1991] provide a summary of the approaches as follows:

Table 3.1: Summary of Research Approaches [Iivari, 1991]

<table>
<thead>
<tr>
<th>Constructive Research Methods</th>
<th>Nomothetic Research Methods</th>
<th>Idiographic Research Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual development</td>
<td>Formal mathematical analysis</td>
<td>Case studies</td>
</tr>
<tr>
<td>Technical development</td>
<td>Experiments; laboratory and field</td>
<td>Action research</td>
</tr>
<tr>
<td></td>
<td>Field studies and surveys</td>
<td></td>
</tr>
</tbody>
</table>

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 application of study in an industry environment would seem to suggest that the constructive research approach would not be wise, as the social environment would seem to obviate the need to create artificial structures to capture the data.

Iivari [1991] bases the definitions of both nomothetic and idiographic research approaches on the differentiation of approaches discussed by Burrell and Morgan [1979]. Nomothetic research is concerned with exploring empirical data in order to test hypotheses of a general character about phenomena studied. Nomothetic research is concerned with a search for, and evidence to support, general laws or theories that will cover a whole class of cases. Such research emphasises systematic protocols and hypothesis testing within the scientific tradition. It has already been alluded that every business will face a slightly different environment and set of drivers. The creation of any truly generic approach will be so generic as to have little or no value
to a specific business. It is also positivist in aspect, although less positivist than the constructive approach, and is not in keeping with the interpretative approach employed here. Therefore the nomothetic approach, although initially appealing, should not be used in this instance.

Finally, the idiographic research approach is concerned with exploring particular cases or events and providing the richest picture of what transpires. The aim is to understand a phenomenon in its own, particular, context – the essence of interpretivism. Idiographic research emphasizes the analysis of subjective accounts based on participation or close association with everyday events [Cornford and Smithson, 1996]. This seems to be the best approach to employ, given the constraints. The research is most definitely participative, and the subjectiveness is also assumed, under the interpretivist philosophy. However, with the research approach decided, the different kinds of methodologies need to be considered to form a research strategy.

3.5 Research Strategy

There have been many papers published dealing with approaches to research strategy [Galliers, 1992], [Alavi, 1994], [Remenyi and Williams, 1996]. Each of these papers demonstrates different strategies that can be employed for researching. Yin [1989] provides a useful summary table for empirical studies based on interpreting real world events (reproduced below, Table 3.2):
Table 3.2: Relevant Situations for different Research Strategies [Yin, 1989]

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Form of Research Question</th>
<th>Requires Control Over Behavioural Events?</th>
<th>Focuses on Contemporary Events?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment</td>
<td>How, why</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Survey</td>
<td>Who, what, where, how many, how much</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Archival analysis (e.g. economic study)</td>
<td>Who, what, where, how many, how much</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>History</td>
<td>How, why</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Case study</td>
<td>How, why</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

However, Järvinen [2001] finds Yin’s approach somewhat limited, especially when considered in conjunction with Iivari’s [1991] definition of research strategies. Järvinen proposes that the structure of categories of strategy groupings, in line with Iivari’s [1991] definitions, should be based on particular classifying principles, disaggregating approaches in a hierarchical manner, shown in Figure 3.1:

![Figure 3.1: Adapted from Järvinen's [2001] Taxonomy of Research Methods](image-url)
Mathematical research strategies attempt to prove or disprove a theorem, lemma or assertion. The management of an IT portfolio is not a mathematical concept, nor does it rest upon a theory, lemma or assertion. Therefore this category should not and will not be used.

Conceptual analytical strategies study the assumptions behind constructs. This involves the analysis of theories, models and frameworks that have been used previously in other empirical studies, and then results are generalised. The management of a portfolio of IT systems in this case, is a specific instantiation, not a generalised application.

Theory testing uses such methods as laboratory experiments, some types of case studies [Lee, 1989], field studies, some longitudinal analysis methods, and field testing. With respect to the management of a portfolio of IT systems, it could be that these strategies could be employed, as the management could be considered a highly specific case study.

The final grouping of strategies, the theory-creating strategies, seems to include everything the other strategy groupings have not: the normal case studies (as defined by [Yin, 1989]), ethnographic methods [Hammersley, 1990], grounded theory [Strauss and Corbin, 1990], [Glaser and Strauss, 1967], phenomenography, contextualism [Pettigrew, 1985], discourse analysis, action research, participant observation, phenomenological studies, and others. It is this category that seems best fitted to the research direction, the management of an IT portfolio.

Several key approaches are seen to be relevant to the particular instance. These are contextualism, grounded theory, case studies, action research, and participant observation. These will now be discussed in turn, and the final research strategy chosen.
3.5.1 Contextualism

Schönp [1983] stated that practitioners have degrees of constancy and variation in their patterns of working. The constants are:

- The media, languages and frameworks that the practitioners use to describe reality and conduct experiments
- The appreciative systems they bring to the problem definition, to inquiry development and evaluation, and reflective conversation
- The overarching theories by which they make sense of phenomena
- The role frames within which they set their tasks and define boundaries.

Pettigrew [1985] found many parallels between Schönp's work and the contextualist approach to research. The principle aspect of this is the situational nature of the inquiry. Pettigrew [1985] maintains that one of the core requirements of a contextualist analysis is to understand the emergent, situational, and holistic features of an organism or process within context, rather than prepare a model of only some of the variables.

Järvinen [2001] states the basic principles of contextualism to be:

1. The mutual nature of inquiry
2. The balance between the researcher's involvement and distance
3. The notion that knowledge is created through a process of making rather than discovered through a process of knowing
4. The importance of the situational and multifaceted character of meanings in research settings
5. The holistic study of emergent processes in particular and changing contexts.

Pettigrew [1985] advises researchers contemplating using contextualist approaches to move into an organisation, use multiple data gathering techniques, and act as a consultant, principally by giving feedback to the organisation.

It can be seen that there are several similarities between the ideal contextualist approach, and the approach that could be used within a business study, such as
performed in this thesis. A contextualist element must therefore be included when analysing the qualitative data.

3.5.2 Grounded theory

Grounded theory is a pragmatic approach to the analysis of qualitative data. Glaser and Strauss [1967] stress that research is conducted in markedly different settings, by researchers who will inevitably impose very different personal and professional qualities on the study. They argue that it would not be feasible to impose a single set of methodological rules to be followed on all occasions. It has been further argued that any imposition of this kind would be detrimental to the quality of research – 'a standardization (sic) of methods (swallowed whole, taken seriously) would only constrain and stifle social researchers' best efforts.' [Strauss, 1987]. Grounded theory is intended by the authors to present guidelines and rules of thumb [Glaser and Strauss, 1967].

Strauss [1987] opines that the analysis of qualitative data should be geared towards the generation of new concepts and theories. This is in line with the broad categorisation put forward by Järvinen, discussed above. It therefore centres on how the researcher makes use of qualitative data, and emphasises that qualitative data, unlike some quantitative data, cannot be left to ‘speak for itself.’ Both Glaser and Strauss [1967] and Strauss [1987] continually emphasise that theories should 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. Denscombe [2002] makes the example that a researcher employing grounded theory should not set out to test a theory, but should instead approach the situation without a rigid set of ideas that shape what he or she focuses on during the investigation. Strauss [1987] equates this with setting out on a voyage of discovery. Interestingly, this corresponds very closely with the lessons of Suzuki-roshi Shunryu Daioshi [1970], who insisted that all things must be approached ‘with the beginner’s mind’ and that understanding comes from the act of understanding, having ‘looked at what is through the eyes of a beginner.’
Given the fact that no assumptions are made, the selection of people, instances, and tools to be used will reflect the developing nature of the research and cannot be predicted at the start. The 'voyage of discovery' will force the researcher to iteratively create, implement, and analyse new angles of investigation and enquiry. This lack of assumptions makes defining sample sizes somewhat difficult, and so the concept of theoretical saturation was coined [Strauss, 1987]. Theoretical saturation means that the researcher should go on sampling and analysing until additional analysis no longer reveals anything that contributes towards a greater understanding of the situation.

Layder [1993] states that it is the flexibility of interpretation of ideas across a broad range of research approaches that yields the significant impact of grounded theory; the use of grounded theory must, by its nature, be viewed flexibly, and not followed slavishly. Denscombe [2002] provides a counter argument, by saying that the term ‘grounded theory’ has been devalued, with many researchers using it without paying due attention to the ideals of Glaser and Strauss.

With that said, grounded theory seems very apt to employ in industry, where it is extremely hard, if not impossible to state exactly what is going to occur in advance, as it depends very much on numerous variables, politics not least of which. Thus a grounded theory approach would be viable for this type of project.

3.5.3 Case Studies

The Benbasat et al [1987] paper provides a comprehensive set of definitions and suggestions for the conduct of case research. 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 are defined in various ways, and a standard does not seem to exist. However, a definition compiled from a number of sources [Stone, 1978], [Benbasat, 1984], [Yin, 1989], [Bonoma, 1985] and [Kaplan, 1985] in Benbasat et al [1987], is as follows.

"A case study examines a phenomenon in its natural setting, employing multiple methods of data collection to gather information from one or a few entities (people, groups or organisations). The boundaries of the phenomenon are not clearly evident at the outset of the research and no experimental control or manipulation is used."

When deciding whether to use the case study approach or not, there are a number of factors to consider. If there is a need to focus on contemporary events or phenomena in a natural setting, clearly the case study is advantageous. The same is also true if there is no strong theoretical base for the research, i.e. if it is a theory building research project, "A rich and natural setting can be fertile ground for generating theories" [Benbasat, et al, 1987]. However, if there were a need for control or manipulation of variables, then the case study would not be appropriate. Within the case study approach there are a number of variations. A key feature of the design of case study research is the number of case studies that can be included in a project. Generally speaking it is better, i.e. more valid and generalisable, to include multiple cases; though there are instances where a single case is instructive [Lee, 1989]. Exploratory studies are generally better served by single cases, i.e. where there is no previous theory. A single case can also be used to test an existing, well-formed theory.

Case studies require multiple data collection methods, the results of which hopefully converge, in order to establish construct validity. Yin [1989] identifies these methods as including:

- Direct observation of activities and phenomena and their environment;
- Indirect observation or measurement of process-related phenomena;
• Interviews – (structured or unstructured);
• Documentation, such as written, printed or electronic information about the company and its operations (also newspaper cuttings);
• Records and charts about previous use of technology relevant to the case.

A case study approach would be apt, as the situation does focus on contemporary events, and could well be a theory-building exercise, and the boundaries are not immediately clear. It would be a single case study, if this is the chosen method, and multiple data sources would have to be considered. Records and charts are of course available within an industry setting, as is the relevant documentation. Observation and interviews are also easily arranged within an industry setting, once the appropriate permissions have been gained.

3.5.4 Action Research

Action research, sometimes incorrectly referred to as practitioner research, is normally associated with small-scale, ‘hands-on’ research projects [Denscombe, 2002]. It has its origins in the work of social scientists in the 1940’s, who advocated a closer tie between social theory and the solving of social problems. It has been extensively used in areas such as organisational development, education, health, and social care. Rapoport [1970] defines a generic action research as “the method which aims to contribute both to the practical concerns of people in an immediate problematic situation and to goals of social science by joint collaboration within a mutually acceptable ethical framework.”

Denscombe [2002] provides four defining characteristics of action research:

• Practical – aimed at dealing with real-world problems and issues, typically in work and organisational settings
• Change – 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 – the research involves a feedback loop to itself, in which initial findings generate possibilities for future investigation
• Participation – practitioners are crucial to the research program, and their participation is active not passive.

It becomes obvious that action research is less of a method and more of a strategy for social research. It does not specify the constraints by which data can or cannot be gathered, nor does is specify the analysis to be performed. Indeed, it seems to borrow from grounded theory a pragmatism in approach, allowing a wide variety of different data and analyses to be performed [Denscombe, 2002] [Susman and Evered, 1978].

Given that the process of research and action are so closely linked, action research focuses very much on the practitioner, it is not practitioner research. Edwards and Talbot [1994] give a very good definition of practitioner research, stating that practitioner research can only be defined as action research when the professional conducting the research engages in frequent reviews and self-reflection on aspects of their own practice as they engage in that practice.

Action research differs from several other approaches in that it is the researcher themselves that is the agent for change [Boutilier et al, 1997], often in partnership with others. The nature of the partnership can take many forms. Kemmis and McTaggart [1988] state that all actors involved in the research process are equal participants and must be involved at every stage of the research. However, there are cases where the responsibility lies solely with the action researcher [Elliott, 1991]. Most action research lies somewhere between these two extremes [Denscombe, 2002], with the aim being the democratisation of the research process. This democratisation relies fundamentally on the respect of the practitioner’s knowledge by the actors in the research [Zuber-Skerritt, 1996].

The wide spectrum of possible partnerships has lead to the development of three types of action research [adapted from Zuber-Skerritt, 1996]:

1. Technical – aims to improve effectiveness of managerial practice. The actors are co-opted and depend on the researcher as a facilitator.
2. Practical – as technical, but also aims at the actors’ understanding and professional development.
3. Emancipating – as practical, but also achieving transformation and change within the existing boundaries and conditions, as well as changing the system itself of those conditions which impede desired improvement in the system/organisation

Denscombe [2002] provides a good summary of the advantages and disadvantages of action research, in Table 3.3:
Table 3.3: Summary Table of Action Research (Denscombe [2002])

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Addresses practical problems in a positive way</td>
<td>• Involvement of practitioner limits scope and scale of the research</td>
</tr>
<tr>
<td>• Personal benefits for the practitioner – professional self-development</td>
<td>• ‘Work-site’ approach affects representativeness of findings</td>
</tr>
<tr>
<td>• Continuous cycle of development and change via on-site research, which</td>
<td>• Integration of research with practice limits feasibility of exercising</td>
</tr>
<tr>
<td>has benefits for the organisation</td>
<td>controls over factors of relevance to the research</td>
</tr>
<tr>
<td>• Involves participation in research from many actors</td>
<td>• The nature of the research is constrained by what is permissible and ethical</td>
</tr>
<tr>
<td></td>
<td>within the workplace setting</td>
</tr>
<tr>
<td></td>
<td>• Ownership of the research process becomes contestable within the framework</td>
</tr>
<tr>
<td></td>
<td>of the partnership</td>
</tr>
<tr>
<td></td>
<td>• The researcher is unlikely to be impartial and detached in the approach to</td>
</tr>
<tr>
<td></td>
<td>the research (a direct contrast to the positivist approaches)</td>
</tr>
</tbody>
</table>

3.5.5 Participant Observation

Becker and Geer [1957] give a concise summary of participant observation, and what distinguishes it from ordinary, systematic observation:

"By participant observation we mean the method in which the observer participates in the daily life of the people under study, either openly in the role of researcher or covertly in some disguised role, observing
things that happen, listening to what is said, and questioning people, over some length of time.”

The intention of participant observation is to preserve the naturalness of the setting [Denscombe, 2002], thereby minimising disruption and seeing things as they normally occur, unaffected by the act of measuring. It seems this is a partial step towards positivity, in that it acknowledges that the observer is a part of the system, but then attempts to minimise that part.

The key aspect of participant observer is access to insider information. The system under observation has aspects that can only be understood through the application of privileged information – insider information. The driver behind participant observation has been anthropology, with an emphasis placed on a holistic understanding of the cultures and events from the point of view of the insider.

The advantages and disadvantages of participant observation are shown in Table 3.4 (adapted from Denscombe [2002]):
Table 3.4: Summary of Participant Observation (Denscombe [2002])

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Basic equipment – requires very little other than the observer</td>
<td>❖ Access – limited options open to the researcher about which roles to adopt or settings to participate in</td>
</tr>
<tr>
<td>• Non-interference – stands a better chance of retaining the naturalness of the setting than other methods</td>
<td>❖ Commitment – very demanding method in terms of personal commitment and resources</td>
</tr>
<tr>
<td>• Insights – provides a good platform for gaining rich insights into the social processes and is suited to dealing with complex realities</td>
<td>❖ Reliability – dependence on the researcher and on the notes leads to a lack of verifiable data, so reliability is open to doubt</td>
</tr>
<tr>
<td>• Holistic – offers holistic explanations incorporating the relationships between various factors</td>
<td>❖ Representativeness – problems with generalising from the research</td>
</tr>
<tr>
<td>• Subjects’ points of view – very good at getting actors’ meanings as they see them</td>
<td>❖ Deception – ethical problems resulting from the lack of consent of actors when the covert method is used</td>
</tr>
</tbody>
</table>

3.6 Conducting the Research – The Choice of Research Strategy and Approach

Having reviewed the various research approaches and strategies available, the researcher has found that the best strategy for use within Rolls-Royce is one of active participant observation. The researcher intends to fulfil an active role regarding Rolls-Royce Naval Marine’s IT portfolio management, whilst at the same time documenting the results of the action and the effectiveness of the tools. The participation is overt, with the management acknowledging the researcher’s
commitment to Loughborough University. This overt participation will avoid ethical issues associated with the covert method, and consent of the individuals and the company itself. With respect to the toolset create and evaluation, a grounded theoretical approach is seen to offer the most advantages, as it allows the business requirements to shape the evolution of the toolset.

With this in mind, the researcher has also been able to split the study period (three years) into three principle components. These will be discussed in the next section.

3.7 Definition of the Stages

As mentioned above, grounded theory is a voyage of discovery, involving an iterative approach to the situation, in a sense, letting the situation drive the solution. The nature of the participant observation meant that the period of study has been divided by the researcher into three stages. The stages are defined both by the nature of the work, and the style in which it was conducted.

The first stage is termed "Introduction to the workplace." It was during this stage that initial observations were made by the researcher, background research was conducted, and various tentative ideas introduced to the workplace. Most significantly, it showed the researcher the necessity of establishing professional credibility within the workplace, enabling transferral of ideas and methods. Activities for this stage are shown in Figure 3.2. The need to establish professional credibility is not unique to Rolls-Royce. Any hierarchical organisation will require some proof of ability before entrusting resource. In a sense, the researcher would be subjected to a 'cost-benefit analysis' by the management of Rolls-Royce.

Professional credibility is vital to the management of a portfolio of IT systems. With the researcher in the role of change agent, it was necessary that the workforce be able to accept ideas from the researcher, without rancour or nervousness. The second stage introduced more concepts and ideas into the workplace, primarily on specific aspects of the business, demonstrating that the researcher had sufficient skills and knowledge in the subject area. Skills such as business analysis, background knowledge of the relevant area, programming knowledge, data analysis, networking, and consensus
Chapter 3

Methodological Review

development skills all had to be demonstrated within the specific business context, and are shown in Figure 3.3. The second stage was a pilot stage for establishing professional credibility, the tools and methodologies being applied on a small number of systems within a single department. This also represented an opportunity for a first pass with the tools and methodologies, leading to valuable feedback before widespread application. This is in line with a grounded theoretical approach, with the researcher acting as both change agent and observing the work place and the effect of the tools/methodologies.

The third stage of the research is that of full participation and wider application of the tools and methodologies employed. This is the stage where the researcher was accepted and valued as a member of the team, and was recognised as a change driver, and a valuable resource to be used. This enabled the researcher to have a significant influence over how Rolls-Royce managed its portfolio of IT systems. It was during this stage that the researcher was involved in many dedicated teams involved in the management of the portfolio, as is shown by Figure 3.4. The development of the dedicated teams increased the speed at which feedback could be passed to the researcher, thus decreasing the iterative cycle time for the creation and development of newer tools/methodologies. All of the stages will be discussed in more detail in Chapters Four, Five and Six.
### Chapter 3 Methodological Review

#### Figure 3.2: Gantt Chart Representation of Activities for Stage One

<table>
<thead>
<tr>
<th>ID</th>
<th>Activity</th>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Period 1 - Introduction to the Workplace</td>
<td>J F M A M J J A S O N D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>Situational research</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>Research scope stories</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>Company research</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td>Expert judgment deployed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.6</td>
<td>Feature analysis - original model</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.7</td>
<td>Research lifecycle analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.8</td>
<td>Identified need for tool for achieving consensus</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.9</td>
<td>Researched Delphi technique</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.10</td>
<td>Identified issues with US</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.11</td>
<td>Research methods to mitigate US</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Chapter 3: Methodological Review

<table>
<thead>
<tr>
<th>ID</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Use of business analysis skills</td>
</tr>
<tr>
<td>2.2</td>
<td>Modified feature analysis employed</td>
</tr>
<tr>
<td>2.3</td>
<td>Identified need to model data requirements</td>
</tr>
<tr>
<td>2.4</td>
<td>Searc stories employed</td>
</tr>
<tr>
<td>2.5</td>
<td>Analysis of methods available for LIS</td>
</tr>
<tr>
<td>2.6</td>
<td>Identified need to collect requirements pre-SSADM</td>
</tr>
<tr>
<td>2.7</td>
<td>Identified need for method to pinpoint problems</td>
</tr>
<tr>
<td>2.8</td>
<td>Identified problems outside of feature analysis</td>
</tr>
<tr>
<td>2.9</td>
<td>Lifecycle analysis on hardware</td>
</tr>
<tr>
<td>2.10</td>
<td>Statistical analysis of defect data employed</td>
</tr>
<tr>
<td>2.11</td>
<td>Identified need to show summary of defect data</td>
</tr>
<tr>
<td>2.12</td>
<td>Hierarchical defect analysis developed</td>
</tr>
<tr>
<td>2.13</td>
<td>Questionnaires researched and developed</td>
</tr>
<tr>
<td>2.14</td>
<td>Questionnaires deployed</td>
</tr>
<tr>
<td>2.15</td>
<td>Cultural problems with change identified</td>
</tr>
<tr>
<td>2.16</td>
<td>Software quality models identified and developed</td>
</tr>
<tr>
<td>2.17</td>
<td>Software quality models bought in by group</td>
</tr>
<tr>
<td>2.18</td>
<td>Cost models examined</td>
</tr>
<tr>
<td>2.19</td>
<td>Identify need to manage stakeholders more effectively</td>
</tr>
</tbody>
</table>

**Figure 3.3:** Gantt Chart Representation of Activities for Stage Two

<table>
<thead>
<tr>
<th>ID</th>
<th>Activity</th>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>System1</td>
<td>2 &amp; 3</td>
<td>4 &amp; 5</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[diagram of Gantt chart showing activities for Stage Two]
<table>
<thead>
<tr>
<th>ID</th>
<th>Activity</th>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Period 3 - Full Participation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>Modified feature analysis (MFA) employed on all systems</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.2</td>
<td>MFA - strategic view employed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>MFA - process view employed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>Entity relationship diagrams considered (actn. 2.3)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.5</td>
<td>Data requirements modelling - modified ERD (2.3)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.6</td>
<td>Identify need to model data flows as well</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.7</td>
<td>Contextual data flow diagrams considered (actn. 3.6)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.8</td>
<td>Simplified data flow modelling (3.6)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.9</td>
<td>Quick architecture analysis employed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.10</td>
<td>Create Quadrant stakeholder analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.11</td>
<td>Modify Quadrant stakeholder analysis for more detail</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.12</td>
<td>Database function mapping need identified</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.13</td>
<td>Database function map used (derivative of 3.9)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.14</td>
<td>Payback decided on use by group</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.15</td>
<td>Cost benefit analysis used</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.16</td>
<td>PDM Options team</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.17</td>
<td>Strategy Development team</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.18</td>
<td>Future business capability group</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.19</td>
<td>Database rationalisation team</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.20</td>
<td>CM/Access rationalisation team</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3.4:** Gantt Chart Representation of Activities for Stage Three
3.8 Summary

This chapter has reviewed the research approaches and strategies available, and the researcher has selected the most appropriate to the situation facing the researcher, that of a grounded theoretic approach to active participant observation. The advantages that this offers, such as letting the business requirements evolve the tools, rather than having the researcher impose his own views on the development, and the flexibility that comes with the participant observation, have lead to this choice. The participant observation is overt, thus avoiding many difficult ethical problems presented by working within an environment but not being subject to the same restrictions as the work colleagues.

The chapter then very briefly introduced the breakdown of the various stages of study, and shows at a very high level, how each of the stages relate to each other and the work performed for Rolls-Royce by the researcher. A more detailed discussion, looking at the ramifications of each stage, how they came about, what questions were asked, and what answers were developed, will be conducted in the following chapters.

The stages of study built on each other, and all were required in order to fulfil the aim of the study, to create and evaluate a tool set for IT portfolio management. The tools will be developed through a process of evolution, driven by business requirements. Although the business requirements will be driven by Rolls-Royce, the researcher is confident that many of the business requirements will be applicable across many companies. The ethos of IT expenditure has shifted from conspicuous consumption to transparent, cost effective, business-integrated systems. Management requires rapid payback from systems, and expect cost savings/efficiency savings. With rising expectations, management have also become better at measuring the IT impact on the business footprint. Economic conditions have also lead to concentration on costs across the business. Rolls-Royce business requirements, a few specialist requirements aside, will be very similar to most large companies. The toolset developed, although specific to Rolls-Royce, will have applications beyond the company, the process of creation can equally be applied to other companies if necessary.
References:


Chapter 3 Methodological Review


Hegel, G.W.F [1821] *Grundlinen der Philosophie des Rechts oder Naturrecht und Staatswissenschaft im Grundrisse* (Foundations of the Philosophy of Right: Natural Right and the Political Science in Outline), University of Heidelberg. Translation by Inwood, M. Penguin, London


Chapter 3

Methodological Review

Pettigrew, A.M. [1985] “Contextualist research: A natural way to link theory and practice” in Lawler, E., Mohrman, A., Mohrman, S., Ledford, G., Cummings, T (eds.) Doing research that is useful for theory and practice, Jossey-Bass, San-Francisco


Yin, R.K. [1989] Case study research: Design and Methods, Sage Publications, Beverly Hills, California

Chapter 4: Period One – Introduction to the Workplace

4.1 Introduction

As introduced in chapter three, this is the reporting of the first of the stages of research. This stage is primarily about background research, both of the subject area, and of the company itself. Each company will have a set way of doing things, set procedures and processes, and its own history. All of these must be learned, and evaluated, so that the researcher can participate in the workplace just as any other employee would. The background research allows the right questions to be asked. It can also help frame some possible answers.

Rolls-Royce products have very long lifetimes (often measured in decades) and consequent product support provision requires long-lifetime systems. This gives rise to a tendency to accumulate aging systems more than would otherwise be expected. These aging systems are often termed ‘legacy systems’ in Rolls-Royce, although this is not completely true in light of the definition of a legacy system in chapter two. However, it is one of the definitions mentioned. This software accumulation presents Rolls-Royce with a high degree of maintenance burden, and therefore the company is very interested in adopting the best possible, cost-effective IT portfolio management strategy. Managing the legacy systems is seen as the most significant aspect of the portfolio management, and consequently, many of the following chapters concentrate on this issue.

Given this business requirement, the aim of the thesis (to create and develop a toolset for IT portfolio management) would not be fulfilled if the researcher did not review options for managing legacy systems. This chapter also sets the basis for fulfilling the first and second objectives. The first objective will be fulfilled through the development of the tool cited in section 4.4. The second objective is fulfilled throughout the entire chapter, as the chapter itself is examining what is currently available and looking at ways in which it could be applied to Rolls-Royce.
4.2 Overview

Figure 4.1: Diagrammatic Representation of Stage One Activities

As mentioned in Chapter Three, Stage One nominally lasted eight months, although the boundary between Stage One and Stage Two is blurred. The majority of activities associated with Stage One are of a research nature.

4.3 Situational (Background) Research

The first activity (ID number 1.1) is that of situational (background) research. This task has two objectives. The first task is to familiarise the researcher with relevant information to the business environment – for example, the specific quality management systems employed. The second task is to collect and collate available research on what has been published on various aspects of corporate IT, such as ERP implementations, generic IT implementations, software metrics programmes, corporate governance with respect to IT/IS, published lessons learned from various projects that have involved IT, and so on. Once the background knowledge is acquired, it is tied into the specific situation at Rolls-Royce, creating situational research.

4.4 ‘Scare Story’ Development

Part of this situational research revealed a large number of ‘scare stories.’ These were published accounts of problems that had been encountered by organisations utilising IT. It was decided by the researcher, following consultation with business representatives, to start collecting some of these ‘scare stories’ for later analysis and application of lessons learned from these. Consequently, this activity (ID 1.2) occupied a significant number of months, and required refreshing during the course of
the research (as shown in figures 5.1 and 6.1). A collection of these scare stories can be found in Appendix Three. The scare stories were analysed by the researcher for lessons learned, and these could then be distributed to the relevant business personnel. Scare stories were the first tool to be developed as part of the research, stemming from a need to illustrate the pros and cons of certain approaches.

4.4.1 Application of Scare Stories

The initial application was fairly primitive, merely distributing the relevant articles that described the story. However, comments from the people on the distribution list lead to the first development, which was to précis the article for swifter reading, pulling out the relevant points. This was much appreciated by those on the distribution list. However, the impact of this was still small. It was evident that many people, especially the high level managers, had little time available to read articles, and produce the necessary analysis and conclusions themselves. To mitigate this, the researcher then started to produce a précis that not only summarised the articles, but also included lessons learned. This was much more acceptable, especially when presented to the management. It required very little time to read and the conclusions were immediately obvious.

The disadvantage of this method is that the lessons learned are defined by one person. This means that the tool is open to bias. However, the tool is an illustrative aid only, and therefore the threat of bias was ignored, the tool is to highlight problems with certain approaches. It guides, and does not dictate.
4.4.2 Scare Story Example

The first instance of using the scare story was simple. An email would be sent out with a link to the relevant article:

![Article about IT Disasters](image)

Figure 4.2: Example of a Link

As can be seen from Figure 4.2, information about the article is not readily available, and requires the reader to click on the link and read the entire article, creating their own lessons learned in the process. Consultation with various members of the distribution list lead to the next version of the scare stories, one in which the shortcut would be sent out with a short précis of the article, as shown in Figure 4.3:

![Article about IT Disasters](image)

- This article describes a collection of IT disasters
- Documented from industry sources
- Published on.....

Figure 4.3: Example of a Link and Précis

The final evolution did away with the shortcut entirely, and merely made it available in a shared directory for those with the time and inclination to view it. Instead, the situation was described, and the lessons learned extracted, together with highlights. Below is an example of an article published by Cushing, 2002, treated in this fashion. The original article is 3 pages long, on a broadsheet spread:
• Published: Computer Weekly, February 2002 by Karl Cushing

Background
• Pilkington, a European glass manufacturer, has a very fragmented and haphazard IT architecture
• Consists of many different legacy systems.
• Seeking efficiency gains, Pilkington launched an ambitious project to integrate as many of these legacy systems as possible.
• Instigated an extensive change-management exercise
• Chose SAP R/3 to centralise planning and scheduling of float glass production throughout Europe, replacing the various legacy IT systems with SAP modules covering finance, fixed assets, purchasing, inventory management, warehouse management, and production planning

Lessons Learned:
• Made a point of utilising both SAP consultants and internal business staff to ensure business ownership of the implementation.
• The team drove through the initial planning, corresponding with all the major stakeholders across Europe.
• Communication between business units, and different levels of management, was seen as being critical.
• Phased roll-out by geographical area, over a two-year period, allowed Pilkington to modify the installation to allow for local requirements.

(Continued on next page)
• Flexibility was offset by the fact that the modifications had to flow back into previous implementations, making each succeeding roll-out more time-consuming.
• Involved significant process change
• Implementation cost Pilkington £21m.
• Payback expected by 2003, due to the operational cost savings

Highlights:
• Value of communication and corporate involvement, in particular when creating an enterprise-wide strategy for resolving a legacy system problem.
• The communication also extended to the implementation phase, where the flexible approach made more work for the implementation team, but significantly contributed to the success of the project, as the business units obviously felt a sense of ownership of the implementation.
• Importance of senior management support of the project is clear

Figure 4.4: Example of an Emailed Précis

The structure allows for quick absorption of the relevant facts, and can illustrate what the pros and/or cons of a given approach are.
4.4.3 Scare Story Summary

Table 4.1 shows a summary of the use of scare stories within Rolls-Royce Naval Marine. It shows the advantages and disadvantages that the researcher has found in the use of this tool, in the formats shown above.

<table>
<thead>
<tr>
<th>Name:</th>
<th>Scare Stories; Lessons Learned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool for identifying portfolio change options</td>
</tr>
</tbody>
</table>
| Purpose:                    | • To gather industry experience of technologies and approaches  
                             | • To distribute lessons learned from these approaches  
                             | • To prevent repeats of past disasters               |
| Reason for Development:     | Research indicated repetition of disasters; the same mistakes were being repeated again and again |
| Format:                     | Textual, with some high level analysis              |
| Requirements:               | Research tool (such as Reuters Business Briefing™, or LexisNexis™), word processor |
| Strengths:                  | • Powerful tool for distributing lessons learned  
                             | • Uses shock to stimulate attention from management  
                             | • Can help avoid established pitfalls                |
| Weaknesses:                 | • You get out what you put in – the analysis can be as detailed as the researcher wishes  
                             | • The analysis is subjective; the lessons learned are what the researcher defines (this could be mitigated with group analysis)  
                             | • Does not offer a way forward, merely signposts what to avoid |
| Suitable for Generic Application?: | The scare stories are specific to certain aspects, and must be refreshed periodically. However, the process is suitable for generic use |
4.5 Company Research

Company research (ID 1.3 on the Gantt Chart, Figure 4.1) was also highly important for this stage. For full participation, the researcher must be aware of, and be bound by, the company processes, procedures, and standards. It was during this time that the researcher had to become conversant in the myriad processes embodied in the Rolls-Royce Quality Management System (detailing the high level processes the govern Rolls-Royce), the Local Operating Procedures, the relationship between the Local Operating Procedures and the Global Quality Procedures contained within the Quality Management System, and the various national and international standards to which Rolls-Royce operated. These standards include various ISO standards, most particularly ISO 9000 family and ISO 14000 family, but also non-ISO standards, such as those for Configuration Management (CM II) and Product Lifecycle Management (PLCS, although this is becoming an ISO standard).

Every company also has a litany of abbreviations and acronyms. In order for full participation, the researcher had to be able to know about these abbreviations and acronyms, and understand what was meant by them and how they related to the product. Significantly, there was also a great deal of research on the nature of the product required. The product is the Nuclear Steam Raising Plant. This was entirely outside of the researcher’s experience, and therefore required a great deal of research to understand what Rolls-Royce was doing with the product, allowing the researcher to become one of the team, sharing understanding.

4.6 Lifecycle Costing

It was during the course of this company research that the researcher was introduced to the concept of lifecycle costing (ID 1.6 in Figure 4.1). Nuclear plants are significantly capital-intensive over a long life span, and so both the company and the customer are very interested in reducing the long term cost of the product. Consequently Rolls-Royce employs lifecycle analysis to help with this goal.

Life cycle costing “includes all costs associated with research, development, procurement, operation, logistical support, and disposal of an individual system and/or
equipment including the total supporting infrastructure that plans, manages, and executes that system and/or equipment program over its full life. [Kirkpatrick, 2002]"

Life cycle costing uses aspects of accounting measures (covered in the Cost Models Appendix) and combines them with aspects of the system/equipment definition to create a probabilistic assessment of the system/equipment’s cost throughout its whole life. One significant aspect of lifecycle costing is the affect of risk and uncertainty. Given the nature of predictions, and the types of data that can be used in lifecycle costing, the output is very much affected by uncertainty, and it is common practice to apply ranges to predictions.

There has been much research into lifecycle costing, but of significant impact is the work by Fabrycky and Blanchard [Fabrycky and Blanchard, 1991]. In a series of engineering studies, Fabrycky and Blanchard showed that approximately 65% of the lifecycle cost is committed by the end of the preliminary design phase. Figure 4.5 shows a graph illustrating the ease of change, costs incurred, domain knowledge, and commitment levels throughout the life cycle. The S curve indicates why it is important to get requirements right early on in the development of an engineering concept. This cost curve is also applicable to non-engineering concepts [Fabrycky and Blanchard, 1991] [Bradley, 1999].
This had two ramifications for the study. First that lifecycle analysis can be employed to minimise the cost of ownership of the IT hardware infrastructure. A significant element of expenditure of an IT project, whether it be development or maintenance [Bradley, 1999], will be on the hardware component. Given the rapid pace of change in the IT sector, this aspect is a continuing problem. It is also two-edged. Not only is there the obsolescence problem, of older assets not being able to run the latest version of an application, but there is also the reverse issue – some applications may be business critical and be unable to run on the newer platform. Change management for this is essential and inescapable.

The life of the hardware is critical – knowing when items of hardware will fail will allow a more efficient roll-out of the newer assets. Weibull analysis on failure data is commonly used for this purpose (see the software metrics appendix for description).

The particular value of lifecycle costing is the output. The output is a cost figure (albeit surrounded by uncertainty) that is intuitively accessible to managers and users alike. It allows an immediate view on the critical question of “how much is this going...
to cost?" It incorporates the entire cost, not just the implementation cost, and therefore allows a clearer picture of the true cost.

The second ramification stems from the cost curve illustrated in Figure 4.5. This stresses the importance of gathering requirements, and developing these requirements, at an early stage. Late fixes are costly, as shown in Figure 4.5, both in terms of budget and allocated time. It is easier, quicker, and cheaper to deal with problems earlier on.

Lifecycle costing was trialled during the second phase of study (ID 2.9 in Figure 4.1), where it was used to develop a cost benefit case for obtaining new servers, rather than repairing the existing ones. Following the successful trial, it was used on a wider scale within the Nuclear business site to develop business cases for all the IT architecture that the Support department was responsible for. The experiences from those uses are documented here for clarity's sake. Table 4.2 summarises the experiences of lifecycle costing.
Chapter 4  
Period One – Introduction to the Workplace

Table 4.2: Lifecycle Costing Summary

<table>
<thead>
<tr>
<th>Name:</th>
<th>Lifecycle Costing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Tool for monitoring/understanding status of IT portfolio</td>
</tr>
<tr>
<td>Purpose:</td>
<td>• Provide a cash price for an item throughout the entire lifecycle of the product</td>
</tr>
<tr>
<td></td>
<td>• Avoids the ‘maintenance’ trap, where cost of maintenance was not taken account of in development</td>
</tr>
<tr>
<td>Reason for Development:</td>
<td>• To allow hardware maintenance costs to be incorporated in the change management process</td>
</tr>
<tr>
<td>Format:</td>
<td>Statistical, with a significant mathematical component</td>
</tr>
<tr>
<td>Requirements:</td>
<td>Detailed hardware definition, including failure rates; mathematical analysis package (such as SuperSmith Weibull™, or Minitab™), infrastructure knowledge</td>
</tr>
<tr>
<td>Strengths:</td>
<td>• Proven statistical process</td>
</tr>
<tr>
<td></td>
<td>• Employed in other areas of the business, so accepted</td>
</tr>
<tr>
<td></td>
<td>• Provides a cash figure for the life of the product, encapsulating the cost of maintenance</td>
</tr>
<tr>
<td></td>
<td>• Immediate impact of £ figure with management</td>
</tr>
<tr>
<td>Weaknesses:</td>
<td>• Highly statistical (sometimes viewed with distrust)</td>
</tr>
<tr>
<td></td>
<td>• Output quality directly linked to input data quality</td>
</tr>
<tr>
<td></td>
<td>• Difficult to explain to people that are not familiar with the process</td>
</tr>
<tr>
<td>Suitable for Generic Application?:</td>
<td>Yes</td>
</tr>
</tbody>
</table>

4.7 Expert Judgement

It was apparent early on that some form of estimation of effort and cost would have to be conducted. The management of a portfolio of IT systems inevitably requires implementation of new software applications. However, the decision to embark on the development of new software should not be taken lightly. Careful scoping out of the problem and requirements must happen. Once this has been done, the company can then work out how feasible the option is, contrasting costs with benefits. However, the costs and benefits of development are in the future. Some form of estimation is required for costs, efforts and requirements. There are many methodologies for estimating costs, effort, and requirements, but there is the question
regarding their accuracy and reliability. One of the most common is to use experts in the field to generate estimates. How accurate are experts?

It has been shown that effort estimation using various methodologies, such as case-based reasoning [Shepperd and Schofield, 1997], rule induction [Hughes, 1996], and artificial neural networks [Lederer and Prasad, 1998] [Mair, et al, 2000], have been shown to underperform human experts. Shepperd and Kadoda [Shepperd and Kadoda, 2001] have found significant problems with stepwise regression, rule induction, case-based reasoning, and artificial neural nets, particularly with respect to false positive predictions.

Effort prediction systems can be shown to lack the ability to supplant human judgement as a best guess of effort. Given the current technology and capability, it is unlikely that this will change [Matson, et al, 1994]. In an interview, Tom DeMarco [Myers, 2001] disclosed that slack is a practical necessity when planning for IT projects that have used complex effort estimation, as even the most complex effort estimation process still does not map to reality in an accurate fashion.

Using formal processes, typically mathematical, to create estimation has many positive characteristics. Critical factors are quickly identified, and can be managed in an appropriate fashion. Experts quite frequently do not seem to have a defined method for achieving their estimates, relying on experience and 'hunches.' However, development occurs in real life, and real life is subject to an enormous number of variables. These variables are extremely hard to account for in a structured, inflexible mathematical model, whereas an expert will take these factors into account unconsciously.

Expert judgement is no better and no worse than using more formal models, and has the advantage of usually being quicker to achieve than the formal models, requiring only a short meeting to gain the estimate. Because of this, as a rule, expert judgement
was employed throughout the study, with a few exceptions. These will be discussed in later chapters.

One significant problem of expert judgement is the matter of the experts themselves. Who is an expert? In Rolls-Royce, there are designated subject matter experts. Having designated people for certain subjects helps tremendously. However, if theirs is the only opinion used, then the estimate stands a significant chance of bias, either conscious or unconscious due to lack of data. This was recognised by the researcher, who tried to mitigate this possibility by gaining the opinions of a number of experts. This then lead to a problem of achieving consensus. After several problems with disparate results, the researcher used the Delphi technique to achieve consensus amongst the experts. The use of this technique will be discussed in 4.10 (ID 1.8 in Figure 4.1).

4.7.1 Application of Expert Judgement

Expert judgement was sought at all points, and was used extensively. Rolls-Royce designated subject matter experts were used, and they were able to identify other expert opinions. For most estimates, a number of opinions were sought, varying from three to eleven, depending on the subject and the number and availability of experts.

The experts submitted their estimations in a variety of formats. During the first stage, the experts were quizzed separately, which lead to the collection of disparate estimates, and lead to the research and use of consensus building methods (ID 1.8 in Figure 4.1).
4.7.2 Expert Judgement Summary

Table 4.1 shows the researcher's experience of using expert judgement within this stage of research at Rolls-Royce Naval Marine.

Table 4.3: Summary Table of the Use of Expert Judgement

<table>
<thead>
<tr>
<th>Name: Expert Judgement</th>
<th>Type: Tool for identifying portfolio change options and specifying portfolio changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose:</td>
<td>To gain estimates of variables such as cost and effort for IT/IS development</td>
</tr>
<tr>
<td>Reason for Development:</td>
<td>Requirement to have costs and benefits defined</td>
</tr>
<tr>
<td></td>
<td>Needed indication of impact on the bottom line</td>
</tr>
<tr>
<td>Format:</td>
<td>Oral; meetings when requirements and estimates discussed</td>
</tr>
<tr>
<td>Requirements:</td>
<td>A group of experts</td>
</tr>
<tr>
<td>Strengths:</td>
<td>Estimates are no better or worse than more formal methods</td>
</tr>
<tr>
<td></td>
<td>Easy to obtain (if have access to experts)</td>
</tr>
<tr>
<td></td>
<td>Results easier to get buy-in from management if estimates from a known expert</td>
</tr>
<tr>
<td>Weaknesses:</td>
<td>Not very accurate</td>
</tr>
<tr>
<td></td>
<td>Subject to expert bias</td>
</tr>
<tr>
<td></td>
<td>Difficult to justify</td>
</tr>
<tr>
<td></td>
<td>Experts not always easy to identify</td>
</tr>
<tr>
<td></td>
<td>Some experts aren't</td>
</tr>
</tbody>
</table>

Suitable for Generic Application?: Yes
Chapter 4

Period One – Introduction to the Workplace

4.8 Feature Analysis

The core of feature analysis, as it is intended to be used, is described by Kitchenham et al [1997]. However, it is proposed that the feature analysis tool be extended to cover other aspects of business requirements, and not be used merely to compare system capability.

The goal of this tool is to avoid the “features-and-functions” battle observed by Hyland [2002] – namely that the system with the most features and functions wins the evaluation. The original feature analysis methodology is presented in a series of articles by Kitchenham et al [Kitchenham, 1996a] [ibid, 1996b] [ibid, 1996c] [Kitchenham and Jones, 1997a] [ibid, 1997b] [ibid, 1997b] [Sadler and Kitchenham, 1996]. In this series, Kitchenham et al describe a generic feature analysis tool. As described in [Kitchenham, 1996a], the reason for the analysis is either to select a tool to carry out a specific function, or to undertake a market survey to establish product capability and availability.

Kitchenham describes the four main roles in an evaluation exercise:

- The sponsor – sets the context, identifies constraints, provides resources, uses the evaluation report
- The evaluator – identifies the candidate method/tools, organises the implementation of the evaluation exercise, analyses the data, and prepares the report
- The technology users – the user population
- Method/tool assessors – responsible for scoring the features

It is noted that some of these roles will be held by the same person. The features are compiled in an iterative fashion, with complexity of the features being left up to the evaluator to decide.

Kitchenham [1996a] describes two sorts of feature – those that can be answered with a simple YES/NO, nominal scale, and compound features that have answers of degrees, and are assessed on an ordinal scale. Simple features can be accompanied by an importance aspect, with suggested scale points ranging from mandatory to nice to
have. Compound features raise the issue of conformance. Conformance recognises the distinct gradation due to the nature of compound features.

The analysis of the feature analysis data is covered by Kitchenham in [Kitchenham, 1996b]. Systems are compared against a specified set of features. Presentation is through the use of graphical techniques. Kitchenham also allows for the possibility of analysis without importance criteria, but notes that this requires several assumptions that can have major effects on the analysis.

4.8.1 Implementation of Feature Analysis

During the first period of study, it was immediately apparent that some form of method would be required to compare the existing systems to proposed systems, and that capability would have to be compared and contrasted (ID 1.5, Figure 4.1). Comparing features between the existing and planned seemed an obvious step.

Feature analysis was trialled on a single system, the configuration management system. The Nuclear business site of Rolls-Royce had an existing system, PIMS (Parts Information Management System), for this function. A new system was corporately espoused, and there was considerable pressure to adopt this new system. However, before the new system could be adopted, the management needed to be absolutely positive that Rolls-Royce could meet all its configuration management contract requirements from the new system.

Compilation of the features list was an arduous task. The first stage of compilation used the existing development records of PIMS. Ten years previously, requirements for a configuration management system had been identified and documented, in line with ISO 9001 requirements. Subsequent changes due to business requirements had also been documented. These were all extracted from the records, and compiled into the features list.

The second source of information was the Rolls-Royce Quality Management System and associated Local Operating Procedures. These detailed what Rolls-Royce was expecting from various activities, configuration management cardinal amongst them.
These clarified the existing features, and added some new ones, based on corporate Rolls-Royce requirements, not specific to the Nuclear business site. The process maps contained within the Quality Management System helped determine what data should be available, when, and where.

The third source of information came from the contracts themselves. The Nuclear business site has numerous configuration management-related contracts. These contracts set out what Rolls-Royce is liable for, and what the requirements of the deliverables are.

The final source of information was the users themselves. These were the final arbiters of the feature analysis list. They were the ones who decided inclusion, and level of feature. The initial features list was circulated to nominated business representatives. The researcher had no control over the nominations – nominations were by the Head of Engineering for Support. These generally took the form of the Chief Engineer for each department, plus one other. The list was circulated, and invitations for comment sent out. The researcher also conducted many meetings with the nominated business representatives to drive out requirements.

A significant step taken by the researcher was to include representatives for the future business from the start. The work was conducted in the Support area of the Nuclear business site, with the responsibility for future business being entirely invested in an completely different part of the business. Including this area of the business helped ensure business continuity, and that the chosen system would be suitable for future business as well as current business.

However, just as with expert judgement, and for the same reasons, gaining consensus on what was important proved very difficult indeed. This added impetus to the search for some form of consensus building approach (ID 1.8 in Figure 4.1). The two forms of output from this tool are shown in Figure 4.6 and Figure 4.7.
Figure 4.6: Example of Original Feature Analysis Tabular Output

The graphical output proved very popular, as it allowed very quick comprehension of the problem. The percentage value on the y axis marks the number of features fulfilled by each system, according to function group. However, the method, as it
stood, did not discriminate enough between features. This did not reflect the reality of the situation. Some features were vital to contract deliverables, whereas others were dismissed by some business representatives as “mere frills.” Consequently, the method was modified during the next period.

4.8.2 Original Model Feature Analysis Summary

The figures above (figures 4.6 and 4.7) can easily be interpreted, especially the graphical representation. For example, in the specific example above, it is obvious that system 2 does not satisfy as many functions as either system 1 or 3, and can therefore be discarded as a viable alternative. System 1 is better than system 3 in all functions with the exception of geometry management and usability. A suggestion that would be made from this graph to the management would be that system 1 would be the best system to have if usability can be significantly improved.

The researcher would commonly issue a report with the information analysed in this fashion, before making a presentation to senior business managers, highlighting the flaws and benefits to each system, and showing how the systems compare against each other.

Table 4.4 shows the researcher’s experience with using the original model of feature requires. The disadvantages in particular prompted the researcher to further develop the model in the next stage of research.
Table 4.4: Original Model Feature Analysis Summary Table

<table>
<thead>
<tr>
<th>Name:</th>
<th>Feature Analysis – original model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Tool for identifying portfolio change options and specifying portfolio changes</td>
</tr>
<tr>
<td>Purpose:</td>
<td>• To compare and contrast features of systems for evaluation purposes</td>
</tr>
</tbody>
</table>
| Reason for Development: | • To allow evaluation between systems  
                        • To ensure contract deliverables to be met  
                        • To ensure legal requirements were met |
| Format:     | Graphical |
| Requirements: | Analysis performed in a spreadsheet, email tool used to circulate feature list, documentation provided explaining feature analysis |
| Strengths:  | • Very intuitive to understand the graphical output; consequently very powerful  
                        • Analysis does not require high levels of technical ability  
                        • Bias introduced by researcher kept to a minimum, as researcher merely records observations  
                        • Holistic – any source of information can be used, any kind of feature can be documented  
                        • Flexible – structure determined by what works, so little in the way of constraints |
| Weaknesses: | • Provides illustration only – it is not a decision tool  
                        • Information gathering is time intensive  
                        • Gaining consensus on what features should be is very difficult  
                        • Can be confusion as to what a feature of a system is  
                        • Loses coherence if too many levels of features are built in |
| Suitable for Generic Application?: | Yes (but features need to be collected every time) |
4.9 Statistical Analysis of Failure Data from IT Systems

As part of the research conducted in this phase, a short review of statistical methods for assessing quality and defects was conducted, (ID 1.7 in Figure 4.1). The derivation of the various methods reviewed are detailed in Appendix Four. The statistical analysis was actually employed during the second phase of study (ID 2.10 in Figure 3.3).

4.10 Consensus Building

Many of the research areas previously discussed found that building a consensus became vitally important when deployed in industry. Consequently, methods of consensus building were researched (ID 1.8 in Figure 4.1). One of the most referenced was the Delphi technique. This technique was also recognised by many in Rolls-Royce, and it was therefore chosen as it could be used without too much difficulty.

The origins of the Delphi technique lie in a process developed by the RAND Corporation in the 1960’s as a forecasting methodology [Cline, 2000]. It was later taken on by the US Government as a group decision-making tool. It was the US Government that formalised the Delphi process.

The Delphi technique, and its associated consensus building, have their roots in the same doctrine – that of Hegelian dialectic of cyclical thesis, antithesis, and synthesis, with synthesis becoming the new thesis [Stuter, 1998]. The central goal of the Hegelian doctrine is a continual evolution to "oneness of mind," a true consensus, with consensus defined as being solidarity of belief. This process means that the Delphi technique fits within the interpretivist approach employed by the researcher. Furthermore, the iterative approach is recognised in grounded theory, and the contextualist viewpoints, both of which are valid in terms of the research methodologies employed by the researcher.

In thesis and antithesis, opinions or views are presented on a subject to establish views and opposing views. In synthesis, opposites are brought together to form the new
thesis. All participants in the process are then to accept ownership of the new thesis and support it, changing their views to align with the new thesis. Through a continual process of evolution, consensus will occur [Mautner, 2002].

Cline [2000] notes that “Delphi is particularly appropriate when decision-making is required in a political or emotional environment.” This would seem to suggest its use within the context of industrial software engineering, as most industrial software engineering is intensely political, with various factions having strongly held beliefs. According to Clare [1994], the specific purpose of the Delphi technique is to obtain consensus from expert individuals concerning problems, future needs or directions. Again, this would seem to suggest its use within the software engineering environment.

Clare gives a high level view of a traditional Delphi questionnaire process [Clare, 1994], as championed by the US Government [Cline, 2000]:

1. Develop the Delphi question (initial broad concern)
2. Select and contact respondents
3. Develop questionnaire #1 and send out
4. Analyse of questionnaire #1
5. Develop questionnaire #2 and send out
6. Analyse of questionnaire #2.
7. Prepare final report.
8. Mail final report to clientele and respondents.

Silverman [2000] has found that there are many other ways to employ the Delphi technique, including focus groups, individual interviews, tele-conferences, and latterly, emails. However, the process of achieving consensus remains; present, collect opinions, collate, analyse, present.

However, it is noted that the formal Delphi technique is fairly long and involved, and requires the presence of a trained Delphi facilitator. For this research, therefore, the technique was modified. One weakness of the Delphi technique seems to be the opacity of the process. The participants are blind to the analysis. In the process outlined above, the analysis and conclusions are developed in seeming isolation from
the participants. This does not aid in the implementation stages, as it is possible for
the participants to feel unrepresented. It can also lead to the suspicions of collusion
with senior management to produce a pre-determined outcome.

To avoid this, the analysis was made more transparent. Simpler analysis was
conducted, and the participants had full view of the analysis and were involved in the
final report.

The rigid process outlined above was made more flexible. Questionnaires were still
used to define the area to be addressed, but use was also made of other avenues of
communication, such as telephony, focus groups, and individual interviews. This was
to achieve greater support from the participants, and give the participants ‘ownership’
of the process. The experience of implementation is covered in further detail in the
next stage of research.

4.11 Identification of Legacy Information Systems

Various definitions of a legacy system have been put forward in Chapter Two. The
researcher chose the non-value system as the most appropriate system, after
discussion with various business representatives.

It was made very apparent, very early on in the course of study, that the management
of Rolls-Royce considered the Nuclear business site to have a serious legacy system
problem (ID 1.10 in Figure 4.1). Consequently, the researcher was obliged to target
research towards that particular aspect of IT portfolio management (ID 1.11 in Figure
4.1). The initial stage was to investigate what possible solutions were available, and
then to match these up with what was feasible at Rolls-Royce. This investigation
straddles both the first and second periods of study.
4.12 Research of Methods to Mitigate Legacy Information Systems

This was the first aspect of investigation into legacy systems - what possible solutions were there (ID 1.11 in Figure 4.1)? The investigation looked at a range of options, from drastic (total replacement) to minimally invasive (user interface modernisation). The results were then tabulated and circulated to relevant business representatives for discussion.

As a parallel activity, a review of systems with respect to the operative definition of a legacy system was conducted. This was to determine the severity of the legacy information system problem at the Nuclear business site. The result was surprising to many people, as it indicated that many of the systems that had been termed legacy, were actually returning value to the company in excess of their cost. The investigation also revealed several applications that were legacy (using the preferred definition), and the results of this study were passed to Rolls-Royce senior management.

4.12.1 Modernisation

This is the alternative to replacement - modifying the existing rather than replacing. Modernisation of an application involves substantial change, whilst still retaining significant portions of the original application. These changes often include system restructuring, new functionality, or new software attributes.

Modernisation is often employed when the legacy system still has substantial business value embedded, and the changes required are more substantial than those that can be implemented during routine maintenance.

Weiderman et al [2000] identified two types of system modernisation, depending upon the level of system understanding required. White box modernisation requires knowledge of the workings of the legacy system in detail. Black box modernisation, on the other hand, requires only knowledge of the external interfaces of the legacy system.
4.12.1.1 White Box Modernisation

This approach requires a detailed understanding of the internal structure of the application. This can sometimes require extensive reverse engineering. Components and relationships are identified and abstracted.

This reverse engineering can be accomplished by domain modelling, extracting information from code and using the results to create abstractions that help in understanding the underlying structure of the system. This is termed program understanding.

4.12.1.2 Black Box Modernisation

Black box modernisation contrasts with white box modernisation in that it treats the legacy system as an opaque object. Only the inputs and outputs have to be examined – the system interfaces represent the majority of the reengineering. This means that black box modernisation is often based around some form of wrapping technique. As Wallnau et al [1997] and Shaw [1995] noted, this generally masks the older systems complexity and removes mismatches between legacy systems and modern systems.

But, as Plakosh et al describe [1999], the disadvantage of this type of modernisation is that it can often require the detailed system understanding granted by white box modernisation methods in order to implement.

4.12.2 Modernisation Methods

There are many methods to solve the legacy system problem. Outlined below are a few of the most common to be considered.

4.12.2.1 Replacement

Replacing legacy systems (also known as the big bang or cold turkey approach) is often useful when the legacy system cannot provide adequate functionality and for which modernisation is not an option. Comella-Dorda et al [2000] state that replacement is normally used with systems that are undocumented, outdated, or not extensible. There are several important risks associated with replacement:
1. Replacement is a redesign process. The system is built from the ground up, and is therefore very resource intensive. These resource requirements could come into conflict with existing requirements if the software is to be developed internally, as the IT resources may be substantially occupied performing maintenance tasks.

2. IT staff designing the new system may not be familiar with the technologies employed to design the new system.

3. The replacement system will require extensive testing. Legacy systems embody critical knowledge, and are well tested and tuned. It is unlikely that the new system will be as robust, or as functional, as the old system. This would evidence itself by a period of degraded functionality with respect to system needs.

The US Department of Defense have, in conjunction with the SEI, issued a set of ten guidelines for DoD Legacy System Migration [Bergey, et al, 1999]. The guidelines stress the importance of having a concrete goal to work to, as scope creep is a cardinal cause of failure. The guidelines also recommend a phased roll-in of the new system in conjunction with a phased removal of the old system. The guidelines also recommend a strong management role in the whole process, but that this role should not override technical realities.

Redevelopment starts from a blank piece of paper. The only carry through from the original system is the data. Everything else is discarded. This is the most complex, time consuming and expensive method for dealing with a legacy system. However, this method has the advantage that everything is updated, and provides the ability to remove unwanted functionality (an evolutionary throwback!), and integrate new functionality.

Bisbal et al [1999] give the example of the Renaissance Project, which re-developed a legacy system on a new platform, using a new language, modern architecture, tools and methods.

Re-development has by far the largest impact on the legacy system. As Bisbal et al note [1999], it will produce the greatest number of changes and throws up some very
difficult problems. The prime problem, however, is always the same – managing the cutover. This is the point at which the old system is replaced by the newly developed system. It is complicated by the importance of maintaining uncorrupted data and the tendency of new systems to have critical bugs.

4.12.2.2 User Interface Modernisation

The user interface is the user-end of the system. It can be the case that the whole system is disadvantaged by a poor user interface. Modernisation of the user interface can often lend a new lease of life to legacy systems.

One common technique of user interface modernisation is screen scraping [Bisbal, et al, 1999] [Carr, 1998]. Legacy system user interfaces are often a series of character-based screens running in a terminal emulator mode. A graphical user interface (GUI) is PC-based and is generally far easier to use than character-based screens. A ‘lighter’ form of GUI can employ a mark-up language, such as HTML, running in a thin client configuration with a Web browser.

One advantage of this approach is that one new user interface can wrap around many legacy systems, providing an apparently seamless database, whilst avoiding problems associated with federating databases. However, a corresponding disadvantage is that it does not address the maintainability of the old system – the system is just as difficult to maintain as it is essentially without change. This drawback is why the methodology is derogatorily known as “whipped cream on roadkill” [Comella-Dorda, et al, 2000].

Application program interfaces can be generated from screen scraping legacy system user interfaces. This technique was applied by Comella-Dorda et al [2000] to a large DoD system integrating an enterprise resource planning (ERP) system.

The critical success path of this option is the design of the user interface. There are many aspects of the user interface that have to be carefully planned. Firstly is the overall layout. Many applications have similar layouts, principally modelled on the Microsoft standard Windows layout: the elements employed, the dialog boxes used,
the consistency of the elements, and how well the affordances map to functions. This gives applications a certain "look and feel," and allows the user to transfer the capability to use applications. The affordances are visual clues as to what something does. For example, the printer icon has a very good affordance for what it does, namely, send things to print. Microsoft is very influential in user interface design, primarily due to the prevalence of their Windows and office products. Consequently, they have bought out many handbooks on the principles of user interface design, [Microsoft, 1999] [Microsoft, 1998].

4.12.2.3 Database Gateway

A database gateway is a specific piece of ‘glueware’ that translates between two or more data access protocols [Altman, et al., 1999]. There are several application- or vendor-specific protocols used, but some have become de facto industry standards:

1. ODBC – Open Database Connectivity: Microsoft’s interface for accessing data in a heterogeneous environment of relational and non-relational database management systems. It is based on the Call Level Interface specification of the SQL Access Group [Microsoft, 1995].

2. JDBC – Java Database Connectivity: defined by Sun for database connectivity between Java applets/applications and SQL databases. The JDBC application program interface defines the Java classes that represent database connections, SQL statements, result sets, and the database metadata [Grage, 2000].

3. ODMG – standard produce by the Object Data Management Group for persistent object storage. It builds upon existing database, object, and programming language standards to simplify object storage and ensure application portability [Barry, 1998].

A database gateway translates specific access protocols into these more standard protocols. Modern platforms and applications typically support at least one of the above protocols, and so data from legacy systems can then be passed into the modern systems.
XML (Extensible Mark-up Language) is a common standard for structuring data and documents for Web compatibility. It is simple text format, derived in part, from the Standardised Generalised Mark-up Language (SGML) [ISO 8879] developed by the World Wide Web Consortium® (W3C).

XML is extremely powerful meta-language for inter-application data exchange, primarily due to its very flexible and extensible method for describing data items and its ability to communicate using the HTTP protocol [WebMethods, 1999]. It is for this reason that XML is finding popularity in business to business (B2B) application integration. The need for cohesive information exchange between disparate systems is the driving force for B2B applications. The flexibility of XML is ideal for this, as domain-specific vocabularies have been developed, and are still emerging.

XML provides formal control over how the text is handled; this can reduce the risk of incorrect input and propagation [Flynn, 1998]. It also allows many different versions to be propagated from a single master with minimal effort – changing the structure, font or formatting becomes simple, without the risk of losing valuable data. Conversion between formats is also nearly eliminated, as the master file format will never change [Adshead, 2000].

As described by Ryback [2002], the XML server acts as a contact point between a corporate infrastructure and the outside world. It communicates with the internal systems through various means, quite possibly using some of the methods presented here. It then interoperates with outside servers through the exchange of XML messages. It is for this reason that XML has emerged as a solution for legacy system problems.

Early XML applications had to have a Document Type Declaration (DTD) developed in order to be well specified. Defining a DTD can require large investments, so as to capture all the necessary mark-ups. However, there are some publicly available DTDs for office documents, memos, industrial applications, technical writing and documentation, general publishing, and military use, and many more DTDs have been
developed and can be bought. The US DoD was one of the first institutions to make large scale use of SGML [Anon, 1998a] [Anon, 1998b] (and its subset, XML).

Recently, the XML standard has been revised to become XML 2.0. In version 2.0, DTDs are no longer required for a properly specified XML document. The move away from the DTD was prompted by weaknesses in the DTD. These shortcomings were [Claßen, 2001]:

1. The syntax of the DTD is different to the rest of the XML document, requiring a different notation and parser
2. There is no way to specify data types and data formats that could be used to automatically map to and from programming languages (given XML’s placement as a translation layer, this was considered a very large problem)
3. There was no common, well-known set of basic elements available, causing considerable fragmentation, and difficulties in finding commonality for transfer of data

Consequently, DTDs have been discarded in version 2.0. The first improvement is that a schema now uses XML, including standard pre-defined and user-specific data types. Defining an element specifies the name and content model, detailing attributes and nested elements. In a Schema, the content model of elements is defined by their type. An XML document adhering to a schema can then only have elements that match the defined types [W3C.org, 2000].

User-defined elements can be formed from the predefined elements using object-oriented concepts of aggregation and inheritance. Aggregation groups a set of existing elements into a new element. Inheritance extends a predefined element so that it could stand in for the original. There are various standards and sub-languages associated with XML, a result of its flexibility. For example, there is a defined XML Schema managing the translation between XML and HTML (Extensible Stylesheet Transformation).

XML communication between databases was employed to great success at the recent Commonwealth Games in Manchester [McTaggart, 2002]. On a budget of less than
10% of that used in the Olympic Games in Sydney, Manchester produced a superior service. XML was used extensively as a medium to communicate between databases. This enterprise application integration approach utilising XML was also noted by IBM in 2000 [Manchester, 2000].

4.12.2.5 Database Replication

This methodology involves copying and maintaining data in multiple databases, in a distributed database system. Changes at one site are captured and stored locally before being forwarded and applied to the centralised depository. This provides users with fast, local access to shared data and greater availability to applications because alternative data access options exist. It also provides failsoft data continuity, as the data is stored at several sites, so if one is lost, the data is still available [Oracle Corp., 2000]. This can reduce the importance of the legacy system by decentralising the data held on that system.

4.12.2.6 CGI Integration

The Common Gateway Interface (CGI) is an industry standard for interfacing external applications with information servers – HTTP or Web servers. Web access is used to interface with the legacy system. This is very similar to screen scraping, except that the CGI script is designed to communicate directly with the core data maintained in the legacy system.

4.12.2.7 Object-Oriented Wrapping

Object technology has been around for several years, but has only recently been employed on a large scale. Object-oriented (O-O) systems can be designed and implemented in such a fashion so as to mirror the business processes they model [Phoenix Group, 2000]. Further techniques, such as encapsulation, abstraction, and inheritance, have aided this uptake.

Object distribution requires a powerful communication mechanism – Distributed Object Technology (DOT). Conventional inter-object communication is not sufficient to pass objects between systems. DOT is a combination of distributed technology and O-O. It uses object middleware to extend object technology into net-centric
information systems. Middleware is defined by Foreman et al. [1997] as "connectivity software that consists of a set of enabling services that allows multiple processes running concurrently on one or more machines to interact across a network." Middleware can take the following forms:

1. **Transaction processing monitors** - provide tools and environment for developing and deploying distributed applications
2. **Remote procedure calls** - enable the logic of an application to be distributed across a network. Programme logic is remotely called as a local routine.
3. **Message-oriented middleware** - provides program-to-program data exchange. Analogous to email in that it is asynchronous and relies upon the recipients to interpret the messages.
4. **Object Request Brokers** - enable the objects that constitute an application to be distributed and shared across heterogeneous networks.

In 1997, the Object Management Group (OMG) defined a standard for the distribution of objects [Object Management Group, 1997]. This was known as CORBA (Common Object Request Broker Architecture). Most object request brokers are CORBA compliant. The most significant of those that aren't, is Microsoft's offering, DCOM (Distributed Component Object Model).

On a process basis, O-O wrapping is a simple concept - individual services are represented as objects; common services are represented by objects; and business data is represented as objects. In practice, O-O wrapping is far from simple [Comella-Dorca et al., 2000]. However, the move from legacy methodologies to the O-O methodology is continuing, and there are many methods and tools available to assist the migration [Cimitile, et al., 1997] [De Lucia, et al., 1997] [Seacord, et al., 1999].

A recent modification of object-wrapping capability is the emergence of the new Java development kit from Sun Microsystems [Udell, 2002] [Koch, 2001]. This release combines the capability of XML as described in 4.2.4, with the O-O advantages. The incorporation of the Simple Application Program Interface for XML (SAX) and the Document Object Model (DOM) have made object wrapping more flexible and easier to implement [Udell, 2002]. The combination of Java and XML has removed many of the difficulties encountered by Comella-Dorda et al. [2000]. As Koch repeatedly
points out [Koch, 2001], this blending of technologies makes O-O far more practicable as a method to leverage legacy system value.

### 4.12.2.8 Component Wrapping

This utilises a very similar process to O-O wrapping, but at a higher level, using a component model rather than an object model. This is a much newer approach than the other alternatives discussed above, but has recently been employed to great effect by Transco [Adshead, 2000].

There are three industry standards applicable to component wrapping:

1. Distributed interNet Architecture™ (DNA) is the Microsoft solution. It is a de facto standard due to the large Microsoft presence in the industry. This is a scaled-up version of DCOM, using the same principles and processes.
2. CORBA 3 component model by OMG. This is the current version of the CORBA standard, and encompasses component as well as object wrapping.

OMG sees Enterprise JavaBeans merging with CORBA 3 to present two dominant standards – Microsoft’s DNA and Sun’s Enterprise JavaBeans. There are numerous advantages of component wrapping for legacy business logic. Firstly, they use supported software and architectures. Secondly, component wrapping provides a radical increase in system flexibility [Comella-Dorda, et al, 2000]. The wrappers can be fully integrated with management facilities and services included with the application server. Lastly, once the functionality of the legacy system is wrapped, each bean can be re-implemented and modernised incrementally, without having to modernise the system in a contiguous, costly effort.

### 4.12.2.9 Net-Centric Computing

This method has evolved to take advantage of the Internet as a “ubiquitous, distributed, platform-independent, peer-to-peer, collaborative computing platform for software applications” [Wiederman, et al, 2000]. Net-centric computing is based upon the use of a distributed environment where applications and data are downloaded from
network server on an as-needed basis. The principle has been in use for decades (UNIX has made use of dumb terminals since its inception), but it has been with the advent of the Internet that this type of architecture has really 'taken off.'

Current practice uses a “thick client/thin server” architecture, where the applications use primarily local resources. Conversely, Sun Microsystems has always been a proponent of the “Network Computer” – a thin client, thick server, but the concept has never really challenged the personal computer dominance [Campbell, 1996]. The emergence of platform independent languages (such as Java), low cost components and high bandwidth connections has seen a renaissance of the concept.

Net-centric computing can leverage existing applications by making them available over the Internet/Intranet. It can also be employed in a variety of ways, ranging from a simple user interface modification (for access using a Web browser), to restructuring the application into an n-tiered set of interacting components. Each component is mobile, and can migrate from server to server – each component has multiple roles. Data can also be similarly mobile.

The International Data Corporation has conducted a series of studies at major corporations [Campbell, 1996]. The studies evidenced a return on investment averaging over 1,000%, with an emphasis on three themes:

1. Rapid deployment on heterogeneous platforms
2. Widespread acceptance and use (primarily due to familiarity with the browser interface)
3. Realisation of openness, linked to the ability to replace components at will.

All of the studies highlighted the influence of a corporate Intranet in being able to take advantage of these new strategies. Some of the companies surveyed were Cadence Design Systems, Inc.; Booz, Allen & Hamilton; Silicon Graphics, Inc; and the Amdahl Corporation.
### 4.12.3 Legacy System Management Options – Summary

Table 4.5 has been adapted from Comella-Dorda et al [2000], and outlines the various approaches, their aims, and their strengths and weaknesses.

<table>
<thead>
<tr>
<th>Legacy Aspect Modernised</th>
<th>Target System Aspect</th>
<th>Strengths</th>
<th>Weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Redevelopment</td>
<td>Anything</td>
<td>Data model</td>
<td>• Future tool support</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Cost</td>
</tr>
<tr>
<td>Screen Scraping</td>
<td>Text-based user interface</td>
<td>Graphical/web-based user interface</td>
<td>• Cost</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Time to market</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Internet support</td>
</tr>
<tr>
<td>Database Gateway</td>
<td>Proprietary access protocol</td>
<td>Standard access protocol</td>
<td>• Cost</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Tool support</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Maintainability</td>
</tr>
<tr>
<td>XML Integration</td>
<td>Proprietary access protocol</td>
<td>XML server</td>
<td>• Flexibility</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Future tool support</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• B2B</td>
</tr>
<tr>
<td>Database replication</td>
<td>Centralised database</td>
<td>Distributed, replicated database</td>
<td>• Performance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Reliability</td>
</tr>
<tr>
<td>CGI Integration</td>
<td>Mainframe data</td>
<td>HTML pages</td>
<td>• Cost</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Internet support</td>
</tr>
<tr>
<td>OOWrapping</td>
<td>Anything</td>
<td>OO model</td>
<td>• Flexibility</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Cost</td>
</tr>
<tr>
<td>Component wrapping</td>
<td>Anything</td>
<td>Component model</td>
<td>• Flexibility</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Integration</td>
</tr>
<tr>
<td>Net-centric computing</td>
<td>Anything</td>
<td>Decentralised database with browser access</td>
<td>• Flexibility</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Cost</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Complexity</td>
</tr>
</tbody>
</table>

This table is a convenient summary of the solutions outlined above. The table itself was used within Rolls-Royce to provide a starting point for proposing possible solutions to the legacy system problem. This process was very effective in provoking discussion as to the way forward. However, at this stage of the study, the business representatives had not placed the full amount of trust in the researcher, and so it was not as useful initially as it became later on in the final stage of study. This implies that a high level of trust and acceptance is required of the researcher before suggestions can be taken at face value, and are not subject to disbelief.
4.13 Summary

This chapter has documented some of the situational (background) research that was necessary in order to start the second, ‘pilot’ stage. It looks at compiling lessons learned logs, and at various management tools that are available and could be put to use in Rolls-Royce in assisting IT portfolio management. It has reviewed possible solutions to a legacy system problem, concentrating on those that would be most applicable to Rolls-Royce.

By doing this, it fulfils, at least part way, the second objective. The implementation and development of scare stories documented in section 4.4 fulfils the first objective, that of compiling and using lessons learned.

The background knowledge is essential in suggesting and putting forward solutions to the IT portfolio management problem that the researcher observed at Rolls-Royce. Some of the tools and methodologies that were used to put these suggestions into practice are covered in the next chapter.
References:


Campbell, I.[1996], The Intranet: Slashing the Cost of Business, International Data Corporation, Framingham, MA.


Grage, P, Introduction to JDBC

[http://shrike.depaul.edu/~pgrage/dist_sys/ds520_p1.htm](http://shrike.depaul.edu/~pgrage/dist_sys/ds520_p1.htm) Accessed 08/00


Koch, M [2001]. “Leverage legacy systems with a blend of XML, XSL, and Java,” *Itworld.com*, 26/01/01


Oracle Corporation Oracle 8i Concepts

http://www.technet.oracle.com/doc/server.805/a58227/ch_repli.htm Accessed 01/01


Phoenix Group, *Legacy Systems Wrapping with Objects*, URL: http://www.phxgrp.com/jowdewp.htm Accessed 01/01


104


Udell, J. [2002] “Java, XML, and Web Services”, *Infoworld*, 25/03/02


Chapter 5: Period Two – Establishing Professional Credibility

5.1 Introduction

The first period, documented in the previous chapter, concerned background research. The second period documents the first time some of these ideas, tools and methodologies were put into practice. This period has two primary objectives. The first is to start the iterative approach to tool and methodology development. The second, and most important, was to establish the researcher’s professional credibility within the company, an essential component which would enable the researcher to place the toolset within the wider context of the Naval Nuclear business, as opposed to a single department.

This second stage represents the first step in actually developing and using a toolset for IT portfolio management within the Rolls-Royce Naval Nuclear business, documenting the initial tools, and how some of them were further developed. This is a key step in fulfilling the aim of the thesis. The second stage is the first time the researcher takes an active role within the Rolls-Royce business, and is the first time in which the grounded theoretic approach of tool development is fully used.

5.2 Overview

Figure 5.1 shows the Gantt chart for this second stage (repeated from Chapter 3). The individual serials are covered in more detail below.

Figure 5.1: Gantt chart for Period Two
5.3 Business Analysis Skills

Business analysis skills are vital to the management of a portfolio of IT products. ID 2.1 in Figure 5.1 shows that business analysis skills that were employed by the researcher on numerous occasions, in a move towards gaining professional credibility. The first occasion was in relation to product data management, both the discipline and the IT tools. The second and third occasions were concurrent, and involved electronic document management and enterprise resource planning respectively. The fourth and fifth occasions involved intranet development and product data management on a wider scale than the first analysis.

The researcher found that business analysis was the term applied by employees to the process of vetting and validating business cases. In all cases, the analysis requires input from several different disciplines. The business cases are presented in what amounts to a balance sheet. Analysing this requires tracing cash flows to their source, and then determining whether the assumptions are valid. Validating these assumptions requires knowledge of the discipline, such as product data management, knowledge of the IT tools and technologies that would be employed to see whether what is being discussed is feasible, and knowledge of the business itself, to see whether the company could benefit from what each IT tool offers, and how.

The researcher wished to become recognised as a point of contact for business case analysis. This has two connotations. Firstly, the researcher is recognised within the company as an expert in various disciplines and is, therefore, included in discussions regarding these topics. The second implication is that the researcher becomes very familiar with various aspects of the business and will be able to know what will and will not work.

Business case analysis was never conducted alone. Most business cases are peer reviewed by the designated experts and then conclusions are drawn. Rolls-Royce employs a multi-stage process to vet and validate business cases in line with ISO 9000 requirements and recommendations.
The skills needed for business case analysis are varied. Financial, IT, and general business knowledge must be combined with communication and presentation skills. The researcher’s past experience had given him such skills and so he was able to put them to full use. When managing an IT portfolio, it is not necessary to possess them, as long as access to them is guaranteed. They also do not need to be embodied in one person. However, achieving consensus in this case would be more difficult, requiring more in the way of consensus building (identified need ID 1.8 in Fig 6.1) than before.

The business analysis was considerably aided by the use of the feature analysis described in section 4.8, with particular reference to the product data management analysis (ID 2.1 – 2.5 in Figures 3.3 and 5.1). It was with this in mind, as well as experience of using feature analysis, that modifications to the method were made, and deployed within the business on numerous systems. This will be described in later sections.

5.4 Application of a Knowledge Base

One of the aspects of business analysis, from the point of view of managing an IT portfolio, is application of financial knowledge coupled with IT knowledge. One of the research activities carried out in the first period of study was developing a background IT knowledge that would be suitable for use within Rolls-Royce. Having gained the necessary knowledge, it was then necessary to be able to impart that knowledge in a fashion that would let other members of the team, and most importantly, Rolls-Royce senior management, understand why the various options had been considered, and why others had been ruled out.

To assist in comprehension, flowcharts were developed, showing what the strengths and weaknesses were for each option considered. This can also be presented in tabular form. These flowcharts constitute a tool that appears in the second stage of the framework discussed in section 5.5. It requires definition of the problem before options can be considered.

The particular example below concerns options for handling legacy systems. This was the most immediate problem facing Rolls-Royce and consequently was the area
in which most options occurred. It was also the area that senior management most wished to get under control and so much effort was devoted to removing and mitigating legacy systems within the Naval Nuclear business site of Rolls-Royce.

![Figure 5.2: Option Summary Flowchart – Database does not reflect business practices](image)

Figure 5.2 shows the first flowchart. The possible options available are shown as light blue boxes. Strengths are green boxes and weaknesses are red ones. The first identified problem is that the database does not reflect business practices. This is mentioned by Lehman and Belady [Lehman and Belady, 1985] and is detailed more fully in Chapter Two. The consequence to the business of not doing anything is to lose business effectiveness.

The next stage of analysis would be to discover the underlying cause of the stated problem. In the flowchart above, two causes are given. Firstly, the database is too fragile to be modified to reflect the changing business requirements. The only option available here is redevelopment. A less severe form of this problem is that the database is difficult, but not impossible, to modify. This presents three possible
options for consideration: redevelopment, component wrapping, and finally object wrapping.

There is a possible fourth option to the specified problem that has been left off the flowchart for clarity. It is possible that the main reason for the lack of modernisation of the database has been a lack of money. The database is capable of being modified, but hasn’t been due to financial constraints. In this case it is possible to use any of the options available and the decision must be determined by the balance of advantages and disadvantages inherent in each of the options.

The same diagrammatic rules apply to the other flowcharts. In Figure 5.3, the identified problem is that the database is hard to connect to other databases, as part of an integration exercise. The cause is that the database has proprietary access protocols. In this situation there are six possible options that should be considered. These are redevelopment, XML integration, component wrapping, creation of a database gateway, object wrapping and net-centric computing. The high level advantages and disadvantages of each option are presented within each box.
Another commonly identified problem dealt with in Figure 5.4 is that users are not happy using the database. A result of this problem can be poor and patchy data and a dependence on information held away from the database. This is bad business practice and does not allow the company to leverage its information assets effectively.

The flowchart identified two causes of this problem. The first is that the database does not reflect business practices. This is covered in a previous figure (Figure 5.2). The second cause is due to difficulties in using the interface. Three possible solutions are available: screen scraping, CGI integration, and redevelopment.
The flow charts are designed to be a very high-level presentation of the various options available for a given scenario. They are not meant to be a detailed exposition of all the benefits and problems associated with each one. They are intended to focus thinking on a reduced set of options, and to closely define the problem and causes to be addressed. More diagrams can be developed as and when required, as more problems are addressed and causes determined.

Table 5.1 shows a summary of the researcher’s experience of applying the knowledge base developed in Stage 1 from situational research.
Table 5.1: Summary of Knowledge Base Application

<table>
<thead>
<tr>
<th>Name:</th>
<th>Knowledge Base Flow Chart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose:</td>
<td>• To show what options are available to mitigate given problems with legacy systems</td>
</tr>
</tbody>
</table>
| Reason for Development: | • Management had placed highest importance on mitigating legacy systems at Rolls-Royce  
• Need to impart reasons for options and why they were being considered/dISCarded |
| Format: | Graphical/tabular |
| Requirements: | Background knowledge, graphical package |
| Strengths: | • A picture is worth a thousand words  
• Aids communication  
• Can be used as a tick list, once problems are identified |
| Weaknesses: | • Need a good deal of background knowledge  
• Sometimes have to go through all options anyway, to allay fears |
| Suitable for Generic Application?: | Yes – but needs maintenance, and is currently only used with respect to legacy information systems. Principle could be applied to any problem |

5.5 Modified Feature Analysis

As described above, the analysis of business cases provided some improvements that could be made to the feature analysis method described in section 4.8. ID 2.2 in Figure 5.1 shows the timeframe during which this occurred, deployed on two systems. The modified feature analysis was employed on PIMS, which enabled a good contrast to the deployment of the original method, which had also been deployed on certain modules of PIMS.

Just as with the original methodology, the first draft of functionality requirements for a ‘dream system’ can be developed from several sources. Firstly, functional requirements can be identified, from development and enhancement notes of the existing systems. Secondly, interviews can be conducted with key business
representatives, users and system developers, using requirements identified in the first stage as a "straw man."

Derived functionality requirements are divided into groups. This can be done at the same time as the derivation of the features, as this might help with the feature extraction. The groups represent several levels of abstraction. At the top level (Level 0), only generic functional topic areas are addressed, such as 'configuration management' and 'change control.' Level 1 requirements included areas such as 'Record Change Control Decisions' and 'Manage Maintenance View.' Level 1 requirements embody more detailed functionality than Level 0. Level 2 requirements embody more detail still. The compiled feature list should look similar to Figure 5.5:

![Feature List Hierarchy Example](image)
Figure 5.5 shows the hierarchical nature of the features, and how the features nest within each other. The highest level feature is the Level 0 feature, and is merely intended to aggregate requirements under a single title, such as usability. Illustrated in Figure 5.5 is only one Level 0 feature. The exact number of Level 0 features will be determined through the application of the feature analysis.

The next stage involves assigning a degree of importance. In the original model, standard weightings are applied to the importance criteria. In a departure from this, the modified methodology allows the users free reign over the relative significance of each requirement. Weightings are added to the Level 1 and Level 2 requirements on a 1–10 scale:

- 1-3: Useful
- 4-7: Important
- 8-10: Essential

The weightings for each requirement are determined from the information provided by the users and developers from the interview rounds, and from user buy-in outside the key business representatives, as the feature sets were sent to users for comment. The feature set now looks similar to Figure 5.6:
Having a classification below "useful" is not considered to be informative, as identified requirements are, by definition, useful. The extent to which each is prioritised is given by its weighting. Level 1 and Level 0 weightings are determined by using the modal average of the requirements that they enclose. Level 1 weightings can also be given by user choice if deemed necessary. However, this option could exaggerate the importance of some Level 2 features. However, changes to the higher level weightings could be permitted if users/developers feel that the modal average does not reflect the importance (or lack of importance) attached to that group.

So far the derivation is the same as the original method, with the exception of the weighting system, which is a slight modification. The more significant modifications
were in compiling different views of feature analysis, regarding systems (the original model), and adding process, strategic intent, and cost views. The different views were developed in order to show how the system's functionality would be mapped to business requirement/strategic requirement or cost item.

These views of modified feature analysis, as developed by the researcher, were applied by a small team of people appointed by the Head of Engineering & Technology. This team was composed of four people:

1) The researcher, who acted as a facilitator
2) An experienced project manager
3) An nominated member of the IT department
4) A user representative

The user representative was one of the designated super-users of the PDM system in the Naval Nuclear business site of Rolls-Royce (PIMS). The team was appointed to compare product data management systems that were available for use. The team spent three months researching the area and then an intensive month compiling the feature analysis. The reminder of the time (three months) was spent communicating the results of the study.

5.5.1 Example Analysis of Feature vs. Systems

Comparing system functionality is the core of feature analysis, as described by the original methodology in section 4.8. It is intended for the evaluation of systems against the set of features. When evaluating the systems, the capability of each system is evaluated against each of the derived, weighted requirements. Several categories of acceptance can be used:

- Yes – the system had functionality that satisfied the requirement
- TBD – To Be Done; the system, at present, does not have that functionality, but there are plans that have already been put in motion to satisfy that requirement
- No – the system does not have the functionality, and there are no plans to implement this functionality in the near future.
The inclusion of the TBD acceptance allows a very quick ‘planned functionality’ measure to be evaluated, allowing a look beyond the status quo. This was as a result of a suggestion by several business representatives that some of the systems investigated were in the process of being developed and, therefore, assessing suitability on the basis of current capability was unfair.

This view is perceived as being very useful as it encapsulates the strengths and weaknesses of each system in a way that is immediately understandable. It is possible to see at a glance why some systems are unsuitable. It is also possible to extract features that are essential and examine whether the systems cater for these requirements.
As noted in section 4.8, the percentage value on the y axis is the percentage of weighted features fulfilled by each systems according to function group. The graphical, percentage-based presentation was remarked on by many of the business representatives as being a very concise manner in which to portray a very complex situation, and thus it was kept throughout the iterations.

5.5.2 Analysis of Feature vs. Process

The feature analysis methodology is extended, referencing requirements to official company processes, as documented in the Quality Management System. This requires a detailed breakdown of the processes to be conducted, to find what data is required at what stage in the process. For companies certified by international quality assurances standards, such as ISO 9000 and related standards, this process deconstruction is relatively straightforward, as the operating processes should be fully documented as part of the certification processes. Difficulties may occur if the processes have not been updated, although again, if the companies have the quality standards fully enforced, this should not be the case. In the case of Rolls-Royce, the processes were well documented and adhered to, as shown by the ISO 9000 certification.
The process view associates features and data with process. This has two advantages. Firstly, it delineates responsibility for certain features. It is common for management to take responsibility for single processes. If this is so, and it can be proven that certain features are required by certain process, then business champions are nominated by default. The second advantage is clearly organising and delineating the role of the systems under evaluation, so it can be shown exactly what processes this system will support and which ones it does not. This aids communication, and prevents unrealistic expectations from being formed.

It was found that this view did not add much detail to the overall picture. This was primarily due to the way in which the Rolls-Royce Business Process Model has been developed – the overwhelming majority of the features were required by most of the main processes.

This particular view was used by the business, but it has not been taken up with the same alacrity as the system comparison view. There are several reasons for this. Firstly, it was shown most of the features are required by most of the main processes. The only useful aspect came at the margin, to see which processes did not use certain features. The summary presentation was in the same format as the system comparison graph (Figure 5.7). Consequently it was difficult to distinguish the differences between the processes. The full listing, an example of which is presented below in Figure 5.9 was more useful. It was found that only Level 2 features could be marked this way, as the answer was of a Yes/No variety, and it proved impossible to wrap up Level 1 and Level 0 features within the same format.
Chapter 5
Period Two – Establishing Professional Credibility

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Highest Level Requirement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Level 1 Requirement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>6</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>5</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>5</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>7</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>3</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>8</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>8</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>6</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>9</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>2</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 1 Requirement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>8</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>7</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>7</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>9</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>9</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>2</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 1 Requirement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>7</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>4</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>9</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>1</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Level 2 Requirement</td>
<td>2</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

Figure 5.9: Feature Analysis, Process View

The benefits of using this view are simple. It creates a very good understanding of the process framework of the business. It requires that features be mapped to the process that requires them. This engenders a large amount of ‘process-awareness’ in the team performing the analysis. This view was useful in creating process awareness in the various teams that used it. However, it has not had the same effect on the business as the system comparison view.

5.5.3 Analysis of Feature vs. Strategic Intent

The feature analysis methodology was expanded to compare and contrast features with strategic intent. This requires the strategic intent to be mapped out to a fine detail. However, the intent of this comparison is to ensure that the strategic intent can be met by a selection of features. It also discloses where capability is missing, so that new features can be documented, and given high priority as being part of the strategic intent. This has similar advantages to the process view. It clearly delineates what is required for which strategic intent, and therefore helps to nominate business champions for the system.
This view provided some interesting information on when improvements had to be scheduled and how vital they were to have in place. The advantages were that it exposed gaps in essential capability that would be required to expand the business in new directions. Figure 5.10 illustrates an example of the summary view – how many features each target requires. As can be seen, this view does not disclose much information. Figure 5.11 shows an example of the more detailed listing, with weightings, and the earliest date each feature is required. This view obviously discloses more information.

The disadvantage was that none of the scheduling was a surprise! The dates for the strategic objectives were set, and unsurprisingly the requirements coincided with these.

![Percentage of Features Required](image)

Figure 5.10: Strategic Intent Summary View
The benefit mainly accrued to the developers, who could use the data to prioritise development work. They knew what requirements had to be enabled by when. However, the graphical outputs were not particularly useful in this respect. It was the matrix and the required features and dates that were far more useful.

### 5.5.4 Analysis of Feature vs. Cost

The cost aspect is intended to inject a financial element to each feature, and thus provide a useful tool for managers to determine exactly where the expenditure is going to be required to develop and deliver a functional capability. This view is intended to help identify ‘quick hits.’ It will allow management to identify areas where rapid improvements can be made, and identify the areas with the greatest benefit to cost ratio. However, on attempted implementation, this view ran into considerable problems.
Firstly, it was difficult to pin exact costs on to specific features. Most features would be bundled, and a cost incurred per bundle. Differentiating this cost was deemed too complex, and would be too artificial to mean anything.

Secondly, there were problems with estimating the costs themselves. When dealing with current capability, this is irrelevant – the costs have been incurred. But for future developments, where this view would have been of most use, estimation of the costs was considerably hampered by not having a good estimation tool. This, in part, lead to ID 2.18 in Fig 5.1, which called for an examination of possible cost models.

5.5.5 Implementation Issues with Modified Feature Analysis

The team went through several iterations to gather features. A memo to the departments pointed out the necessity of gathering good requirements, and returns were mandated by senior management. Five departments were mandated to reply, and all five complied. Gaining senior management support for this was essential to the timely completion of the feature analysis listing. One concern with this method was the selection of representatives from each department. This was not at the discretion of the researcher. The composition of the focus groups that evaluated the feature list for each department was at the discretion of the department heads, who signed off the feature lists as being an accurate representation of their department’s requirements. Because of the nature of the departments, it was not possible to determine the composition of the focus groups in all cases. Where it was possible, the focus group was normally composed of a senior engineer and several junior engineers.

Given that the remit of the study was to look at both current and future capabilities and requirements, the future business requirements had to be collected. Defining these was more difficult. This required a detailed analysis of business strategy and aims. This took considerably longer to define, as it was fraught with uncertainty as to exactly what was required. Many interviews were conducted with key business representatives using an iterative approach.

Time constraints were placed on the team, so the individuals had to be interviewed by a certain date. This meant that some potential interviewees could not participate
because of work commitments. In addition, it was decided to bias the number of interviews in favour of the larger departments, so that the process would be more likely to discover any dissenting views. Most departments had two to four individual interviews each, although one had seven due to its large size.

The interview structure was in two parts. The first part invited the interviewee to explain what they did and what data and capability was required from an IT system point of view. The second part of the interview looked at the features themselves, and tried to match up what they had said in the first half with the features listed. This technique was very successful, with very few deviations from the compiled list. This was interpreted to mean that the feature list was indeed representative of the department as a whole.

Weighting the features was left to the discretion of the users. The researcher proposed weightings suggested by the core team in meetings with representatives of the business units. These were then adjusted up or down as necessary. The modified Delphi technique (discussed in section 6.10) was used extensively. It is possible that the weightings can be adjusted to present one system more favourably than another. However, the weightings were compiled by users, not the researcher of the feature analysis, and all weightings were evaluated by the users themselves. This iterative process mitigated the risk of bias in this respect.

One constraint imposed by the strategic direction was that the number of systems employed by Rolls-Royce should not increase without good reason. The preferred option was to have one single system that could operate with all departments' requirements. This meant that the highest weighting assigned was used for this feature. Logs were kept of the individual business unit requirements and weightings, but for the final analysis, only the highest weightings were used.

Feature analysis has proven to be very successful at Rolls-Royce and is now used in several parts of the business. The methodology is currently being considered for inclusion within the Rolls-Royce project management process. It was originally intended to expand the feature list beyond the second level. However, when this was attempted, it was found that the picture became very complicated and much more
difficult to manage. The power of the technique is in its simplicity, and this was being lost by trying to simplify too much data.

An additional benefit has been found for systems developers. The identified features gives insight into what functionality is important for users, distinguishing between what is needed and frills that excite the users but are not actually necessary. This enables prioritisation and planning of future systems development.
5.5.6 Modified Feature Analysis Summaries

Tables 5.2 to 5.5 show the experience gained in the deployment of the various views of feature analysis.

Table 5.2: Summary of the Use of Modified Feature Analysis – System View

<table>
<thead>
<tr>
<th>Name:</th>
<th>Feature Analysis – system view</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Tool for identify portfolio change options and specifying portfolio changes</td>
</tr>
<tr>
<td>Purpose:</td>
<td>• To compare and contrast features of systems for evaluation purposes</td>
</tr>
</tbody>
</table>
| Reason for Development: | • To allow evaluation between systems  
                          • To ensure contract deliverables to be met  
                          • To ensure legal requirements were met |
| Format:             | Graphical |
| Requirements:       | Analysis performed in a spreadsheet, email tool used to circulate feature list, documentation provided explaining feature analysis |
| Strengths:          | • Very intuitive to understand the graphical output; consequently very powerful  
                          • Analysis does not require high levels of technical ability  
                          • Bias introduced by researcher kept to a minimum, as researcher merely records observations  
                          • Holistic – any source of information can be used, any kind of feature can be documented  
                          • Flexible – structure determined by what works, so little in the way of constraints |
| Weaknesses:         | • Provides illustration only – it is not a decision tool  
                          • Information gathering is time intensive  
                          • Gaining consensus on what features should be is very difficult  
                          • Can be confusion as to what a feature of a system is  
                          • Loses coherence if too many levels of features are built in  
                          • Not very useful if all processes use all features |
| Suitable for Generic Application?: | Yes (but features need to be collected every time) |
### Table 5.3: Summary of the Use of Modified Feature Analysis – Process View

<table>
<thead>
<tr>
<th>Name: Feature Analysis – process view</th>
<th>Type: Tool for identify portfolio change options and specifying portfolio changes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Purpose:</strong></td>
<td>• To compare features with corporate processes</td>
</tr>
<tr>
<td><strong>Reason for Development:</strong></td>
<td>• Links responsibility for processes in with responsibility for aspects of IT</td>
</tr>
<tr>
<td></td>
<td>• ISO 9000 requirement</td>
</tr>
<tr>
<td></td>
<td>• Provides business champions</td>
</tr>
<tr>
<td><strong>Format:</strong></td>
<td>Graphical</td>
</tr>
<tr>
<td><strong>Requirements:</strong></td>
<td>Analysis performed in a spreadsheet, email tool used to circulate feature list, documentation provided explaining feature analysis</td>
</tr>
<tr>
<td><strong>Strengths:</strong></td>
<td>• As for system view</td>
</tr>
<tr>
<td></td>
<td>• Develops a good knowledge of a corporate quality management system</td>
</tr>
<tr>
<td><strong>Weaknesses:</strong></td>
<td>• As for system view</td>
</tr>
<tr>
<td></td>
<td>• Very difficult to use if the processes are complex/numerous</td>
</tr>
<tr>
<td></td>
<td>• Different parts of the business might have different processes – modelling all of these is time consuming</td>
</tr>
<tr>
<td><strong>Suitable for Generic Application?</strong></td>
<td>Yes (but features need to be collected every time)</td>
</tr>
<tr>
<td>Name: Feature Analysis – strategic intent view</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Type:</strong> Tool for identify portfolio change options and specifying portfolio changes</td>
<td></td>
</tr>
<tr>
<td><strong>Purpose:</strong> • To link features with strategic intent</td>
<td></td>
</tr>
<tr>
<td><strong>Reason for Development:</strong> • Pinpoint requirements for future business • To highlight areas where development was needed • To provide reasons for budget lines to be created with specific IT development in mind</td>
<td></td>
</tr>
<tr>
<td><strong>Format:</strong> Graphical</td>
<td></td>
</tr>
<tr>
<td><strong>Requirements:</strong> Analysis performed in a spreadsheet, email tool used to circulate feature list, documentation provided explaining feature analysis</td>
<td></td>
</tr>
<tr>
<td><strong>Strengths:</strong> • As for system view • Provides a very good insight into corporate direction and strategy • Shows exactly where features are required, leading to useful discussions as to how to acquire them • Useful for systems developers</td>
<td></td>
</tr>
<tr>
<td><strong>Weaknesses:</strong> • As for system view • Often difficult to define future requirements if business expansion is outside of current capabilities • Unexpected events can happen, requiring different functionality – forecasting is not an exact science • If intent is not clear, can be drawn into discussions about the strategy, not the IT component of it</td>
<td></td>
</tr>
<tr>
<td><strong>Suitable for Generic Application?:</strong> Yes, but should only be used where the strategic intent is very clear</td>
<td></td>
</tr>
</tbody>
</table>
Table 5.5: Summary of the Use of Modified Feature Analysis – Cost View

<table>
<thead>
<tr>
<th>Name:</th>
<th>Feature Analysis – cost view</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Tool for identify portfolio change options and specifying portfolio changes</td>
</tr>
<tr>
<td>Purpose:</td>
<td>• To link costs to requirements</td>
</tr>
<tr>
<td>Reason for Development:</td>
<td>• Assists in allocation of funding/budget</td>
</tr>
<tr>
<td>Format:</td>
<td>Graphical</td>
</tr>
<tr>
<td>Requirements:</td>
<td>Analysis performed in a spreadsheet, email tool used to circulate feature list, documentation provided explaining feature analysis; meeting rooms for discussion</td>
</tr>
<tr>
<td>Strengths:</td>
<td>• As for system view</td>
</tr>
<tr>
<td>Weaknesses:</td>
<td>• As for system view</td>
</tr>
<tr>
<td></td>
<td>• Very difficult to link most features with costs, as many features come as a bundle</td>
</tr>
<tr>
<td></td>
<td>• Costs difficult to estimate – expert judgement used</td>
</tr>
<tr>
<td>Suitable for Generic Application?:</td>
<td>No – too many weaknesses make the output subject to considerable doubt</td>
</tr>
</tbody>
</table>

5.6 Requirements Capture and Engineering

Requirements are essential for the development and management of IT systems. ID 2.3 and ID 2.6 in Figure 5.1 both mention some sort of requirements capture. To a large extent, the feature analysis method described above captures a large proportion of the functional requirements. However, an information system is far more than merely a programming construct. It has been recognised early on (see Appendices Two and Four) that requirements management is vital to IS management success. Bringing together all of the tools described above into a cohesive framework would enable much more efficient use of time and resources. The researcher wished to be able to present a toolset for IS portfolio management that was an integrated set, rather than just a collection of random tools.

Currently, Rolls-Royce uses the SSADM process to document requirements. However, this process does not stretch back far enough to govern all requirements gathering (See Chapter Two for discussion of SSADM). In practice it starts with a
functional requirements specification and ignores many socio-political aspects of IS development and management.

Consequently another methodology was examined for use, Checkland's Soft Systems Methodology (SSM), on the grounds that it had been employed within Rolls-Royce previously, and people were familiar with the rather more flexible approach it allowed. (See section 2.8 for a discussion of SSM in more detail).

However, even SSM was not flexible enough to encompass what the researcher was trying to accomplish. SSM is more of a project management tool. The researcher was not in a position to formally project manage, and thus many of the aspects of SSM would have been difficult to follow. The researcher needed the flexibility to introduce tools, and drop tools, as and when required. However, some aspects of SSM were very appealing, particularly the holistic approach to identifying variables and the use of rich pictures.

As a result, the researcher created a suitable framework for use within Rolls-Royce as a precursor to the SSADM process, based very loosely upon SSM. It was introduced within Rolls-Royce as a variant of SSM in order to ease uptake. It was the intent of the researcher to continue to add tools to this tool set, within the framework, during the course of study. Tools that have already been used would be incorporated in the framework.

The modified framework has three sections:
1. Problem definition – working out what is wrong
2. Option Development – working out possible ways of fixing what is wrong
3. Strategy Development – working out exactly how to fix the problem

The use of scare stories (see section 4.4) showed that, in several cases, the wrong problem had been addressed when developing a new system and so the IT managers were left with the same problems, and a host of new ones, entailed by new development. Taking lessons learned from this, the researcher decided that the first stage of IT portfolio analysis and resultant change management should be to define the problem carefully. It is most likely that several problems will be defined during
Chapter 5  Period Two – Establishing Professional Credibility

this stage. Tools that are primarily for understanding the current status quo would be used here. An example of this type of tool would be questionnaires and interviews. The output is expected to be some form of report, detailing the problems, and their causes. One aspect of SSM that is carried over to this framework is the use of rich pictures. A rich picture is a pictographic representation of a certain situation. Commonly, they are drawn on a whiteboard, although implementation in Rolls-Royce extended to the use of a graphical package on a laptop coupled to a projector. Rich picturing was a technique that was extensively employed for drilling down to the cause of the problem. A picture was very helpful in working out why things where happening and where inefficiencies were likely to be. A summary of the use of rich pictures is included below in Table 5.6.

The second stage of IT portfolio analysis and resultant change management is that of developing options. Once the problem is defined, there are many possible ways of dealing with it and supplying remedial action. The decision of which option to eventually use is left to the senior management. The sole purpose of this stage is to provide the senior management with as much detail about the options as possible, in a form that is easy to understand and will allow the senior management to make informed decisions. Tools that fall into this category are feature analysis, and business analysis.
### 5.6.1 Rich Picture Summary

Table 5.6 shows a summary of the researcher’s experience when using rich pictures in Rolls-Royce Naval Marine.

#### Table 5.6: Summary of the Use of Rich Pictures

<table>
<thead>
<tr>
<th>Name:</th>
<th>Rich picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool for understanding the IT portfolio and identifying IT portfolio change options</td>
</tr>
<tr>
<td>Purpose:</td>
<td>To show complex situations in a pictographic way</td>
</tr>
</tbody>
</table>
| Reason for Development: | • Useful tool from SSM  
|                 | • Needed a tool to draw out what was going on and show it |
| Format:         | Pictographic |
| Requirements:   | Whiteboard; can be done on computer with graphical packages (such as Microsoft Visio™ or PowerPoint™) |
| Strengths:      | • Extremely effective at provoking discussion  
|                 | • A picture is worth a thousand words |
| Weaknesses:     | • If done on whiteboard, can be difficult to transcribe  
|                 | • Looks better if have some artistic talent!  
|                 | • Caution is required – pictures can also lead to misunderstandings if not presented with a common interpretation |
| Suitable for Generic Application?: | Yes |

### 5.7 Alternative Methods to Analyse Defect Data

ID 2.11 and 2.12 both require defect analysis capability to discharge. Several methods were considered, and are detailed below.

Defect management is an important part of IS management. Maintenance costs can consume significant proportions of the overall budget. When considering a change to the status quo, the maintenance costs must be considered. More importantly, future maintenance costs must also be considered, as it will be these future incurred costs that will determine whether a system is still returning value to the enterprise. There are many statistical tools available for use. A selection of these are detailed in
Appendices One and Four. For use in Rolls-Royce, a simpler tool was sought for several reasons. Firstly, the statistical tools require a significant volume of data before they can be used. In many cases, this data was not available or was of uncertain quality. Secondly, statistical approaches are generally viewed with distrust, so a tool was needed that avoided this problem entirely.

One solution is a hierarchical software defect analysis tool that was developed as a taxonomy to classify the reasons for code change events according to the cause. The purpose is to ascertain the pace of change, and reasons for change. The taxonomy adopts a simple, four-layer, hierarchical approach.

![Hierarchy of Software Defects](image)

The taxonomy is prompted by the cause of the error. The lowest layer, as shown in Figure 5.12, representing the smallest change type is termed 'code.' These are errors caused by mistakes in the code itself.

The second level, that of design, is caused by errors in design. These, of course, are remedied by changes to the code, but the cause was in the design of the program, not in the coding itself. The third level is caused by a fault of the model used. This fault incorporates both of the fault classes below it, as fixing the fault will require changes to both the design and the code of the program.
Chapter 5  Period Two – Establishing Professional Credibility

The final layer is business. This reflects that business needs change. The fault is caused by changing business needs, as reflected in Lehman’s Laws (section 2.4). Changes of this magnitude require changes to the model, design, and code of the program.

An essential pre-requisite of this tool is a documented software change process that details the reasons for a software change. Going through the faults discovered, it will be possible to demonstrate how much the system has changed after being launched.

It is expected that the hierarchy will also hold for the results, with the majority of faults being code-related, followed by design, model, and finally business. This tool is a coarse-grained approach, designed to yield an insight into how well the application is fitted to the business. All results must be taken in context. For example, in a fast-changing business, it is to be expected that there will be many more ‘business’ faults than in a more slowly moving one. However, in some areas, such as payroll, one would not expect payroll requirements to change rapidly over a long time frame.

This tool was developed as a simple measure to pinpoint were defects were being caused. The tool requires identification of all software defects as one of four hierarchical types. The lowest type is code. The software defect has been caused by a mistake in the code itself. The next level is design. The defect has been caused by a fault in the design. The level after that is model – it is the model that is at fault, rather than the design or code. The final level is that of business – the business context has changed, making the model out of date.

This tool was trialled on the Part Identification Management System (PIMS). It was found that the vast majority of software defects reported were due to code problems, as was expected. What was of interest to the strategy development team was the very low number of defects prompted by business changes and model changes. In ten years of operation, less than 10 defects had been prompted by changing business or an inaccurate model. This result prompted an investigation as to why this was so. What emerged was that the original modelling team had been very thorough, and had embodied a large amount of capability and flexibility into the system. This result was
very encouraging, and added great support for keeping PIMS, with modernisation where appropriate. Table 5.7 shows the lessons learned from applying hierarchical defect analysis.

Table 5.7: Summary of Experience Gained from Applying Hierarchical Defect Analysis

<table>
<thead>
<tr>
<th>Name</th>
<th>Hierarchical defect analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool of understanding the existing IT portfolio</td>
</tr>
<tr>
<td>Purpose:</td>
<td>• To show the root causes of changes to existing systems</td>
</tr>
</tbody>
</table>
| Reason for Development:| • Needed to review defect data in a simple fashion  
                             • Needed an indication of how stable business requirements were |
| Format:               | Numeric                      |
| Requirements:         | Defect data, some way of recording numbers in each class |
| Strengths:            | • Very simple  
                             • Easy to understand output  
                             • Needs very little explanation |
| Weaknesses:           | • Very simple – complexities are not explained  
                             • Only provides information, does not come with any decision rules |
| Suitable for Generic Application?: | Yes |

5.8 Graphical Defect Analysis

As mentioned above, a graphical representation of development time was employed as a very simple means of giving the user an intuitive view of the system data. The graphical analysis was conducted on various systems by the PIMS evaluation team and the strategy development team. The graphical analysis was most applicable to the PIMS system, as it was the largest and most complex of the current systems in Rolls-Royce. Below is an example of the output, as it was performed on a module of PIMS.

The application analysis involved portraying which files occupied most of the development time, and also to show which files occupied the greatest amount of developer time per update. These results can be seen in Figure 5.13 and Figure 5.14:
Chapter 5  
Period Two – Establishing Professional Credibility

Proportion of Developer Time Taken Per File

Figure 5.13: Proportion of Developer Time Taken Per File

As Figure 5.13 demonstrates, the majority of developer time was spent on a minority of files. There were 170 files included in the analysis and more than three quarters of developer time was spent on only four of them.

Average Developer Time Spent on Updates For Each File

Figure 5.14: Average Developer Time Spent on Updates for Each File

Enquiries made showed that these files do have a high degree of usage and are amongst some of the most heavily used in the application. However, there are other files that are used equally as much, but do not occupy the same developer time. For example, drg001 is the drawing entry screen, and is the most heavily used file in the module. Yet patches for this file occupy only 4% of total developer time.
Figure 5.14 shows the developer time per update for each file. This leads to an interesting result – there are three files that take a large amount of time per update. The files that appeared in Figure 5.13 are still represented and often they have many patches, but these patches seem to be easier to create and apply than for four of the files in the drg series.

![Total Update Frequency For All Files](image)

Figure 5.15: Total Update Frequency for All Files

Figure 5.15 shows the update frequency for all files of the Drawings module. There are several noticeable peaks. The initial peak occurred when the application went live in 1993. There then followed several peaks as other modules went live or were changed. 1994 and 1995 both have large peaks. 1996 and 1997 saw a fairly constant number of updates until the end of 1997 and the beginning of 1998. A separate peak also occurred near the end of 1998.

The 1995 peaks occur after the introduction by Rolls-Royce of a unique identifier number for drawings. These identifiers were assigned by PIMS. This forced all drawings to be registered with PIMS. This corresponded with a large increase in the number of users, and so more bugs were discovered. It can be seen that the perturbations caused by this decision had an effect throughout 1994 and can be traced to halfway through 1995.

A final peak is noticeable at the end of 1998. This can be explained by the Y2K corrections that had been implemented almost immediately preceding the peak.
Figure 5.16 shows the cumulative number of the system defects reported. This smooths out the peaks noted in Figure 5.15, and reveals the overall trend. This is steadily increasing, which was not unexpected. It is interesting to note that the trend could be described by a straight line, implying a constant level of activity.

A conclusion that can be drawn from Figure 5.15 and Figure 5.16 is that a simple graph can show the maturity of an application quite easily, and can also give a fairly accurate picture of the stability of the application. The peaks in the latter part of the module’s life are explainable through externally imposed requirements/decisions, not through a shortcoming of the code itself. These results can be used to forecast likely developer workload in the future for a baseline, as the number of reported bugs stayed fairly constant for the better part of a year.

The graphical analysis was presented in report format and in presentations to business representatives. However, the level of enthusiasm and understanding of the situation was limited when compared to that of the feature analysis. It was found that simple, pictorial metrics are desirable because they are intuitive, easy to collect and are often available for previous and current projects. The graphical representation of the data can provide a simple, easy to understand illustration of any system trends. They also enable external influences, and their effects, to be identified.
The graphical analysis is ideal for producing simple, quick, and useful portrayals of the current state of affairs. It also makes it easier to spot trends. Although it does not provide any mathematical backing for the spotted trends, it is, nevertheless, useful.
5.8.1 Graphical Defect Analysis Summary

Table 5.8 shows the experience gained from the application of graphical defect analysis.

Table 5.8: Experience of Using Graphical Defect Analysis

<table>
<thead>
<tr>
<th>Name:</th>
<th>Graphical defect analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Tool for understanding the existing IT portfolio</td>
</tr>
<tr>
<td>Purpose:</td>
<td>• To show possible patterns in defect data</td>
</tr>
<tr>
<td>Reason for Development:</td>
<td>• To avoid statistical analysis of defect data</td>
</tr>
<tr>
<td>Format:</td>
<td>Graphical</td>
</tr>
<tr>
<td>Requirements:</td>
<td>Defect data, spreadsheet</td>
</tr>
<tr>
<td>Strengths:</td>
<td>• Patterns can be seen, explanations derived</td>
</tr>
<tr>
<td></td>
<td>• Simple and quick</td>
</tr>
<tr>
<td>Weaknesses:</td>
<td>• Explanations have to be derived, the graphs have no explanatory power</td>
</tr>
<tr>
<td></td>
<td>• Didn’t reveal anything startling</td>
</tr>
<tr>
<td>Suitable for Generic Application?:</td>
<td>Yes, but hierarchical defect analysis more useful</td>
</tr>
</tbody>
</table>

5.9 Statistical Analysis of Failure Data from IT Systems

As part of the study, the researcher subjected some failure data collected from the PLMS system to regression analysis, to determine if there was any detectable pattern. The derivation of the various methods are detailed in Appendix Four. The statistical analysis was employed during the second phase of study (ID 2.10 in Fig 5.1) but the experiences gained are documented here, alongside the research.

The result of the analysis showed that there did not seem to be any detectable pattern to the defects, other than the expected J-curve following the initial implementation. On review of the results with Rolls-Royce management, the researcher encountered a high degree of non-interest in the statistics, verging on disbelief. There was a high degree of scepticism regarding statistics in general, and the researcher would not employ this particular tool again. Table 5.9 summarises the experience.
Table 5.9: Summary of Statistical Analysis of Failure Data

<table>
<thead>
<tr>
<th>Name:</th>
<th>Statistical Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Tool for understanding the existing IT portfolio</td>
</tr>
<tr>
<td>Purpose:</td>
<td>• To illustrate patterns in failure data</td>
</tr>
<tr>
<td>Reason for Development:</td>
<td>• To provide mathematical rigour to ‘eyeballed’ opinions</td>
</tr>
<tr>
<td>Format:</td>
<td>Statistical, with numeric output</td>
</tr>
<tr>
<td>Requirements:</td>
<td>Statistics package, defect/failure data</td>
</tr>
<tr>
<td>Strengths:</td>
<td>• Mathematically rigorous</td>
</tr>
<tr>
<td>Weaknesses:</td>
<td>• Scepticism regarding use of statistics endemic</td>
</tr>
<tr>
<td></td>
<td>• Did not show anything unexpected</td>
</tr>
<tr>
<td></td>
<td>• Not really appropriate for situation</td>
</tr>
<tr>
<td>Suitable for Generic Application?:</td>
<td>No</td>
</tr>
</tbody>
</table>

5.10 Questionnaire Use

Questionnaires are generally more suitable for gathering information on subjective data than objective data [Silverman, 2000]. This is mainly due to the effect of user bias – the people filling in the questionnaire are likely to bias the answers to their own advantage, thus eliminating objectivity. This can be both a conscious and a subconscious desire [Borgatti, 1998].

Questionnaires can be used to collect both qualitative and quantitative data. Quantitative data is more mathematically tractable than quantitative data. Hence there are several methodologies that have been evolved to enable mathematical analysis of qualitative data. Likert scales are proposed to enable this mathematical analysis in this case (See section 5.10.1).

Careful thought is needed for questionnaire construction [Anon., 1997] [Borgatti, 1998] [Silverman, 2000]. Firstly, the objectives of the questionnaires need to be clear – what is being measured and why is it being measured [Anon., 1997]?
Questionnaires (ID 2.13 and 2.14 in Figure 5.1) were used to gather more knowledge about the current situation than could be provided from the analysis of documentation and the creation of the feature list—namely detailed information about application use across the portfolio. The questionnaire used Likert scales to bound most of the questions, but there was an open question at the end of the first questionnaire. This was to solicit opinions from the users outside of the strict boundaries dictated by the Likert scales.

The questionnaire, similar to the example in Appendix Five, was sent to the entire Rolls-Royce database-user population. This entailed sending out more than 100 questionnaires. Approximately 70% of the questionnaires were returned without reminders. On telephone conversations, with non-returnees, most were happy to have the questionnaire read out to them and answer it verbally. Those that couldn’t spare the time generally filled in the questionnaire and sent it back to the researcher. In the end, all questionnaires were accounted for, 11 with no answers. The causes for all 11 non-returns were identified.

The questionnaires allowed the user to remain anonymous, but a name and specialty fields were included. The instructions on the questionnaire and on the accompanying email stated that filling in these fields was entirely voluntary. Most users filled these forms in, and were quite happy to state their opinions, most especially on the last, open, question on the questionnaire. This question provided a lot of information about what people really thought about the systems, and if improvements were made, which areas would represent ‘quick hits.’ They also provided several useful ideas that were incorporated into the option development stage.

A second set of questionnaires was prepared, examining future business requirements and capabilities. The capability questionnaire was emailed to department heads. Intranet submission was considered, but had to be discarded, due to the nature of the networks within the Nuclear business unit of Rolls-Royce. Due to security reasons, Rolls-Royce maintains several separate networks. The department heads are on some of these, but not all. Intranet submission would have necessitated the creation of numerous mirror sites to capture the returns. It was deemed easier to mail the questionnaire to the relevant people.
Unlike the first questionnaire, the reply to the second one was not mandated by senior management. In retrospect, this was a mistake, and the exceedingly high returns on the first questionnaire were not replicated here. It is suggested that the lack of mandatory replies meant that the department heads assigned a lower priority to the completion of the questionnaire than the researcher would have liked. The rate of return was below 25%, and many of the questions had not been answered.

Sampling frames were not applicable, as the entire population of department heads were being queried. Piloting a questionnaire was also ruled out for two reasons. Firstly, there was a time component, as senior management wished the analysis to be performed quickly. Secondly, the target population was small. A pilot questionnaire would have been useful, but it was felt that it would require little extra effort to send out the questionnaire to the entire population, rather than a sample.

As has been mentioned above, the returns were not very informative, as many department heads assigned a reduced priority to this questionnaire. Also, many of them replied that they were following corporate strategies and did not illuminate exactly how they were pursuing them.

The analysis was very simple, as the questions were merely to discover the tactical solutions being adopted to the various corporate strategies. Consequently, there was very little scope for compensating for possible bias.

It is recommended that a different process be followed if this is used again. The questionnaire-based Delphi approach does not work well on very small, disinterested populations. A more interactive approach, such as round-table meetings or teleconferencing would be preferable, as this would ensure that the researcher could collect more opinions in a shorter time frame, and work towards consensus more effectively.
5.10.1 Likert Scales

There are various semantic tools available when constructing questionnaires with qualitative questions. Munshi [1990] reports that the most common in business research are Likert scales, Stapel scales, and the semantic differential. All attempt to provide quantitative solutions to qualitative questions.

The Likert scale was the result of several years' work by a variety of people. The original concept of semantic scales was introduced by Likert [1932]. Likert published an improved version of his semantic scales in a co-authored book, using Likert scales in psychological analysis [Murphy & Likert, 1938].

Critiques by Green and Rao [1970] and Lehmann and Hulbert [1972] have both highlighted the problem, also evinced by Murphy and Likert [1938] that the number of choices employed in the scale has a direct effect on the accuracy and precision of the instrument. However, neither of the two papers provided empirical evidence to demonstrate the problem. Rather, the issue was attacked from a philosophical viewpoint, that qualitative questions ought to have a spectrum of answers, and that the setting of quantitative scales automatically bounds the answer, and can prevent the full expression of the solution.

However, research by psychologists has tended towards a more pragmatic approach to choice provision using Likert scales than would be applied by the purists Green and Rao [1970], and Lehmann and Hulbert [1972]. Miller [1956] found that human beings tend to have an internal process that is discrete, rather than continuous, as implied by Green and Rao [1970], and moreover tend to have a spectrum of seven discrete choices, plus or minus two.

For this reason, the construction of Likert scales tends to aim for a scale of seven, plus or minus two choices. Clearly, the number of choices is also determined by the nature of the question – sometimes it may be necessary to have more than nine choices, or fewer than five.
Likert scales are applied in two fashions for this particular study. The first application is to provide the user opinion of how much they think they use particular applications. This can be compared to the access logs, and will provide information on how much users are aware they use an application. As a subset, a simple Likert scale will be applied to find out which users use what applications. This is not strictly a Likert scale, but the Likert process enables a similar look and feel to the possible answers.

Likert scales can also be used in the construction for questionnaires during the deployment of the Delphi technique.

### 5.10.2 Questionnaires Summary

Table 5.10 summarises the lessons learned from the application of the sets of questionnaires sent out by the researcher.

**Table 5.10: Summary of the Lessons Learned from Questionnaire Application**

<table>
<thead>
<tr>
<th>Name: Questionnaires</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type:</strong> Tool to understand the existing portfolio and identify possible portfolio change options</td>
</tr>
<tr>
<td><strong>Purpose:</strong> • To collect qualitative and quantitative data from a population</td>
</tr>
</tbody>
</table>
| **Reason for Development:** • Needed to elicit user opinions  
  • Interviews with all users impractical |
| **Format:** Text |
| **Requirements:** Distribution system (mail, email), word processor |
| **Strengths:** • Very simple  
  • Easy to deploy  
  • Easy to collect answers  
  • Will provide a snapshot of opinion |
| **Weaknesses:** • Can be hard to construct  
  • Care needed when constructing questions  
  • Piloting does help, but can reduce the sample population  
  • Provides a snapshot, not a commentary  
  • Fails in the face of disinterest – mandated answers can mitigate |
| **Suitable for Generic Application?:** Yes |
5.11 Cultural Problems with Change

ID 2.15 in Figure 5.1 reflects the perennial problem with any change process – ensuring that the user population is supportive of the change. Resistance to change must be taken into account when managing a portfolio of IT systems, as change will be an inevitable part of this management process. The approach taken in this project was to ensure communication between the stakeholders and obtain senior management support at every opportunity.

5.12 Software Quality Models and Cost Models

Many of the tools discussed need some form of cost or effort estimation to be used (ID 2.18 in Figure 5.1). The solution used in Rolls-Royce by the researcher was to use expert opinion. However, it was recognised that there are other alternatives. This was also the case for estimation of quality (ID 2.16 and ID 2.17 in Figure 5.1).

The researcher and several business representatives looked at various cost and effort models, in an examination of whether any could be used within a specific context. Several models were considered, all of which are detailed in Appendix One. All of the models considered were commercial products. This created several problems. Firstly, in order to use them, licenses had to be purchased. This was expensive, especially given the fact that the models were IT-based, and this would incur a separate set-up charge from EDS. Secondly, neither the business representatives nor the researcher were sure which models would be most suitable for Rolls-Royce. Therefore a selection was necessary, exacerbating the first problem.

However, all the models were based on previous models that were freely available. It was decided by the project leader to conduct a quick survey of these models, to ascertain:

- What data was required
- What the output was (confidence)
- Whether the data required was available within Rolls-Royce

The result of this quick survey was that none of the models, even the newer commercial ones, offered a significant advantage over expert judgement, and that the
data required was not often available. The team felt justified, therefore, in not using
detailed cost models.

5.13 Problem: Managing Stakeholders

ID 2.19 (in Figure 5.1) recognises a new problem – that of managing stakeholders and
their expectations. When dealing with small aspects of a single problem, or acting as
an expert point of contact, the management of stakeholders had not been an issue.
With the more active participation in the workplace that was occurring towards the
end of the second stage, the researcher found that the management of stakeholders
was problematic for both himself and the business. Taking this need further, a tool
had to be developed to manage the expectations. This process carried on into the third
phase of study, as better means of managing stakeholders were sought. A solution
was developed, and this is detailed in Chapter Six.

5.14 Definition of 2nd Generation Problems

Part of the option development, although not explicitly recognised on the overview, is
the definition of “2nd generation problems”. These are problems that are incurred
through change. It is all too often the case that change can resolve one set of
problems, only to yield an entirely new set. This problem was recognised by the
researcher and the business teams and so efforts were made to elicit “what if?”
statements. During this phase of study, there was no attempt to generate a tool for
doing this, only in creating an awareness in Rolls-Royce that this could happen and
that ramifications of choices need to be subject to careful consideration.

5.15 Summary

This chapter documents the period during which the researcher established a
professional reputation within the workplace. It is a key step towards the use of a
toolset for proactive IT portfolio management within Rolls-Royce. The chapter
employs much of the situational (background) research conducted in the first stage
(section 4.3), including lessons learned (section 4.4), the development of a structure to
embed background knowledge (sections 4.3 and 5.4). This fulfils the first three
objectives in a pilot stage, and enables the researcher to ‘set the scene’ for use within
a wider Rolls-Royce context (the fourth objective) by establishing a professional credibility.

This chapter has looked at several tools, notably feature analysis and various forms of defect analysis, in order to clarify information available on the current state of the IT portfolio. However, this is admittedly only from the select point of view of a single department within a business unit. It did not represent views from the wider Rolls-Royce Naval Nuclear business site.

It is noted that some tools and methodologies were far more successful than others. Notably successful in the pilot stage were feature analysis and the simplest form of defect analysis, the hierarchical defect analysis. Notably unsuccessful was the statistical analysis of the defects, which came up with a more mathematically rigorous proof of what the hierarchical defect analysis had suggested. However, the researcher found that most of the management were distrustful of statistics in general, and preferred the more straightforward tools and methodology.

This represented the single most important lesson learned by the researcher during the pilot study – the simpler the tool appeared, the more useful it became as a way of nurturing management support. Since management support is a key component of project success (as shown by the scare stories, Appendix Three and section 4.4), this was an important factor in the development of new tools/methodologies in the last period of research.
References:

Anon, *Course notes – Questionnaire Design*, 1997
http://www.cc.gatech.edu/classes/cs6751_97_winter/Topics/quest-design/ Accessed 09/02

Borgatti, S.P. [1998] *Principles of Questionnaire Construction*, Analytech,
http://www.analytictech.com/mb313/princip.htm (accessed 12/02/03)


Likert, R. [1932], “A Technique for the Measurement of Attitudes”, *Archives of Psychology*,140(6), 1932


Chapter 6: Period Three – Full Participation

6.1 Introduction

This chapter documents the culmination of the study, the period during which the aim and objectives are fulfilled. It is the period of study during which the toolset was used by the Rolls-Royce Naval Nuclear business site. It moves beyond the pilot stage documented in chapter five.

The use of the toolset was extensive, and the researcher’s performance during the pilot stage had created a high degree of professional credibility. Consequently various parts of the toolset were used by many teams within the business site. The composition and nature of the teams is discussed below.

This chapter sets the toolset within a structure, loosely derived from a soft systems methodology (SSM), explaining why Checkland’s SSM was selected as a model, and why it was changed to fit the circumstances. The chapter also shows how existing tools, such as entity relationship modelling and contextual data flow diagrams, were modified in a novel way, again in answer to business requirements.

6.2 Overview

The third period of study constitutes the bulk of the study, as can be seen from Figure 6.1.
The third phase of study was full participation in the workforce. It was characterised by inclusion as an acknowledged expert in many strategy teams. The researcher was considered as a valuable resource by the company, just as any other employee. It was also characterised by the continued, iterative development of the toolset, this time within the structured framework that had been considered in the previous phase of research.

More tools were developed and used, and the existing tools were refined, and tested on other systems in other parts of the business, and frequently by people other than the researcher. Feedback from the use of these systems was sent to the researcher, so that modifications could be applied.

The overview shows that the tool set was originally used with respect to product data management within the Nuclear business site of Rolls-Royce, but was then extended outside of this area as the usefulness became apparent.

The sections below cover how the tool set was deployed, and the lessons learnt from deployment. The overall framework of the study is within a grounded theory approach, where iteration and responsiveness to the environment is key. With this in mind, the third stage builds on work done during the previous two phases, in an iterative fashion, developing improvements to the tools, and introducing new ones where necessary.

In the specific instance of product data management, there had been much work performed in the previous two stages of work. This meant that many of the analyses performed earlier were reused when necessary, rather than re-visited, in order to save the time of the participants. The implication of this is that some of the modifications could not be reflected in the analysis of product data management. This reflected the needs of the business, who felt that the area of product data management had been well assessed, whereas the area of electronic document management for example, had not been examined at all, and needed detailed analysis.
6.3 Portfolio Management Framework Review

The third phase of study is characterised by the use of a methodology initially derived from the soft systems methodology discussed in the previous chapter. SSM had already been used within Rolls-Royce, so the concept of using a soft issues-based analytic approach was not new. Traditional software design methodologies, especially SSADM, were firmly ensconced within Rolls-Royce, but it was acknowledged that these tools lacked the ability to capture requirements adequately, and did not have a good way to capture the 'pre-design' ("how can we make the situation better?") and 'non-design' ("what is wrong with the current system?") questions. SSADM in particular is well used in Rolls-Royce in fulfilment of the TickIT certification. However, in the opinion of the researcher, SSADM does not look at the holistic applications of software. The concentration is upon the software itself, not how it is used and how it could be used. It also does not govern the processes which could enforce the use. Consequently, the researcher wanted to use a methodology that could examine these softer issues. SSM does provide this. However, the researcher felt it was not flexible enough to suit the needs discovered at Rolls-Royce. Something was required that was very flexible (can be easily adapted when additional requirements are placed upon the team), but at the same time has a structure that allows sensible analysis. Using SSM as a guide, a three stage process was developed. However, it also pulls in aspects of the Goal Question Metric approach.

The portfolio management framework has three stages:

1. Problem definition
2. Option development
3. Strategy development

Similar to Checkland's SSM, the first stage is problem expression/definition. The first stage requires eliciting requirements, and creating a problem definition that scopes out the cause of the problem, and why it is a problem, and what the ramifications of the problem are. It should be noted that solutions are out of scope at
this stage. The output could be in any format, but a simple statement would be looked for, in preference.

A hypothetical example of this kind of problem definition statement (not taken from experience at Rolls-Royce) could be:

The problem is that a department has failed an ISO 9000 audit, because it is not following processes properly.

It is a problem because the company is committed to excellence, and the ISO 9000 certification is one measure that it employs to show excellence. It is also a problem in that the customer requires the company to have ISO 9000 certification. Losing it could mean losing contracts.

The cause is that the processes do not adequately reflect the required work, following an abrupt change in business, and the process manuals are hard to access and understand, leading to great confusion.

The ramifications, if left unsolved, would be that the company is likely to lose business, and the reputation for excellence will suffer. Mitigating action is strongly urged.

The framework is created to look beyond simple IT systems, and look holistically at information systems. The management of an IT portfolio and resulting change can mean looking beyond IT solutions to process and culture change.

The second stage of the framework melds several of Checkland’s stages together. Checkland had two stages of model development and model assessment. These have been joined under the heading “option development.” This was done primarily
because the researcher found that the business representatives would develop an option and immediately start a critique. To divide the two stages would require culture change and discipline, and the researcher did not feel it was necessary to slavishly follow Checkland's methodology in this instance. This second stage takes the problem definition and proposes many possible options. In practice, this generally takes the form of a brainstorming meeting, followed by a detailed workup. The framework is not intended to have decision rules built in. The responsibility for the decisions rests solely with the management. The tool set is intended to provide the management with information, not to make decisions for them.

Option development is a 'blue-sky' process. It asks the question "what could remedy this problem?" It is an iterative process, whereby a list of possible solutions is generated. This list is then subjected to a quick review to determine whether these options are technically feasible, given the current constraints of the business. Those that aren't, are dropped. The next part then examines the remaining options for financial feasibility. This requires some estimations of cost and benefits to be made, using expert judgement. The aim is to produce a few technically and financially feasible options for the management to be able to choose a single one. The business must be able to explain what these options are intended to accomplish to the management's satisfaction at this stage. A typical output of this stage would be a report of the process, coupled with a meeting with the board, to discuss these options.

The final stage, prior to software development (if required), is that of strategy development. This takes the chosen option and conducts a deep analysis of what exactly is required, and how to best achieve it, given certain constraints. Detailed user requirements and business requirements are necessary, together with a plan for implementation and deployment. This stage examines not just IT but also the processes, so it could be necessary to include process change as part of the strategy development. This stage is not directly addressed by Checkland — his process ends with implementation. This stage in the portfolio management process is probably best described as sitting between Checkland's implementation stage and the previous model assessment stage.
The final stage includes some aspects of the strategy implementation. Ideally, this would be a separate stage. The work will be split into ‘bundles’ which will allowed implementation to proceed quickly, whilst the details that later parts of the implementation are still being worked out. However, it is recognised that dividing line between development and implementation can be very blurred, with implementation beginning whilst the final tactics are being developed. This effect would be increased if work is ‘bundled’ into smaller projects, as is proposed. Consequently, development and implementation are embodied in a single stage. The differences and similarities between the two methodologies can be seen in Figure 6.2.
6.4 Team Creation

The overview in Figure 6.1 shows four teams that the researcher was connected with (IDs 3.16, 3.17, 3.18 and 3.19 in Figure 6.1). These are discussed below. All the teams had access to the tool kit, and kept the researcher appraised of what and how
they were using, plus any comments. The iterative approach, as advocated by grounded theory, was extensively used, and tools/methods updated. The researcher was by this time viewed by the company as a valuable resource, as well as having quickly absorbed a large volume of business-specific knowledge. The combination of the business knowledge and the researcher's skills in utilizing both his knowledge and other people's, and being able to present this knowledge in a variety of useful ways, meant that the researcher was invited to be a member of all these teams by the Head of Engineering. However, time constraints meant that the researcher could only be a member of some of these teams. Consequently, the teams were required to keep the researcher appraised, as mentioned above.

6.4.1 PDM Team Composition

A 'tiger team' was instituted to evaluate PDM options for Rolls-Royce (ID 3.16 in Figure 6.1), in light of the new market conditions highlighted in section 6.4. It was composed of:

1. The researcher, acting as facilitator
2. A senior engineer, representing the current business requirements, and leading the team
3. A senior engineer representing future business requirements
4. A representative from the business improvement department
5. A team champion, from the business improvement department

The responsibility of this team was to examine the business requirements for product data management and recommend a way forward. This was a precursor of the strategy group mentioned below.

6.4.2 Overall Strategy Team Composition

A strategy group was developed as a recommendation of the PDM 'tiger team.' (ID 3.17 in Figure 6.1) This group kept track of progress of all the groups and pulled all the IT infrastructure work together. This team met on a daily basis for three months to create the strategy document for Rolls-Royce. The strategy team composition was:

1. Senior team leader – experienced project manager with years of experience in the business
2. The researcher – facilitator and corporate IT architecture specialist
3. Senior engineer – representing the current business engineering departments, with an interest in CAD
4. Senior engineer – configuration management specialist
5. Engineer – future business data analyst

The strategy development team was also responsible for creating the work plan for implementing the strategy, and for defining the resources and timescales needed to make sure the work plan was accurate. The strategy team was also responsible for the creation of the discrete ‘mini-projects’ as part of the work plan. The initial remit of the team was to finish in the middle of year 3. This would be following the successful implementations of the strategy work bundles. However, the strategy document was kept live, and change was chosen to be on a continuing basis, rather than as a more disruptive step-change process. The team was maintained indefinitely, as shown by the hatching in Figure 6.1, to continue having an overview of the IT infrastructure and related processes of the Nuclear business site of Rolls-Royce.

6.4.3 Future Business Capability Study Team Composition

A future business capability study group (ID 3.18 in Figure 6.1) was put together at the recommendation of the strategy group. The capability group was responsible for detailing the requirements for future business, and feeding them back to the strategy group. The future business capability study group split into two parts. The full team was composed of representatives from all business departments, and the core team. The full team numbered approximately 15 individuals, in addition to the core team. The core team was composed of:

1. Team leader – senior management representative and experienced project manager
2. The researcher – facilitator and co-chair with the team leader
3. Future business representative – experienced engineer
5. Software development team representative – experienced data modeller

The core team met every day to discuss progress, and to make sure that information was being circulated to the full team as necessary. The full team met once a month to
input progress to the core team (if it had not been directed to one of the business representatives between meetings). Just as with the overall strategy group, this group has been indefinitely prolonged, as future business is a continual process.

6.4.4 Database Rationalisation Team Composition

Also at the request of the strategy group, a database rationalisation group was started (ID 3.19 in Figure 6.1). This group was to look principally at how to rationalise configuration management databases within the Nuclear business site of Rolls-Royce. The team numbered seven, with:

1. Team leader – senior engineer and business experience
2. The researcher – provided background knowledge and detailed IT knowledge
3. Software development representative – provided data modelling skills and IT knowledge
4. Key business representatives – one per department that required configuration management databases

This group had a limited remit, as so had a set deliverable and set requirements. The team delivered a report to the strategy group at the end of September of the third year.

6.4.5 Configuration Management Process Rationalisation Team Composition

A further group examined configuration management process rationalisation. This team also passed the conclusions to the strategy team. The team composition was:

1. Team leader
2. Configuration management specialist
3. Business process specialist
4. Business representatives

This team did not involve the researcher, due to time commitments to the other teams. However, all the teams maintained a high level of communication with the strategy team.

All of the teams kept detailed minutes. The proceedings of each team meeting were noted down. In the case of the teams of which the researcher was a member, the minutes were taken by the researcher and emailed to the relevant team members for
comment after each meeting. When the researcher was not a member, the minutes were taken and then emailed to the team members and copied to the researcher.

In addition to the teams mentioned above, there were several ‘offshoots’ that were involved in the creation of the strategy, but were not part of the official strategy development process, as detailed by the teams. These offshoots were temporary teams, set up as a form of brainstorming exercise, widening the breadth of business knowledge that was being input into the various studies. This was part of the communications process, as it was necessary to deliver the intentions of the various teams mentioned above as widely as possible throughout the company.

6.5 Stakeholder Analysis Tool

The management of stakeholders was identified as an issue in the second phase of study, but no tool was then developed that the researcher could introduce to Rolls-Royce. Consequently the researcher developed a tool to designate and help manage stakeholders. This was presented to a small group of business representatives, who suggested modifications and improvements before use.

As part of any requirements capture exercise, identification of stakeholders is a key activity. There are several tools and methodologies available for this process. Several papers have an ‘angle’ on stakeholder analysis [Mason & Mitroff, 1981] [Mitroff & Linstone, 1993]. In industry, the definition of stakeholders tends to be focused on those that have an interest in, and can materially affect, the project.

The tools described by Mason and Mitroff [1981] and Mitroff and Linstone [1993] and the prescribed creation and collation of a list of stakeholders for ISO 9000 certification, do not seem to give the required information in a format that is useful for decision-makers. A tool was required that facilitated the analysis of the stakeholders to determine the levels of interest and influence on the project under review and to determine why these stakeholders have practical influence.

The first stage is the identification of relevant stakeholders. A brainstorming session of the PDM team resulted in a listing of stakeholders relating to product data
management. This list is then divided into quadrants based on influence and interest by the group. An example is shown in Figure 6.3. Each quadrant is identified by a combination of high and low interest and influence. This type of diagram would generally be drawn on a whiteboard, so that names can be moved around easily.

<table>
<thead>
<tr>
<th>High Interest/High Influence</th>
<th>High Interest/Low Influence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stakeholder A</td>
<td>Stakeholder D</td>
</tr>
<tr>
<td>Stakeholder B</td>
<td>Stakeholder E</td>
</tr>
<tr>
<td>Stakeholder C</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Low Interest/High Influence</th>
<th>Low Interest/Low Influence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stakeholder F</td>
<td>Stakeholder H</td>
</tr>
<tr>
<td>Stakeholder G</td>
<td>Stakeholder I</td>
</tr>
<tr>
<td>Stakeholder J</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.3: Example of a Stakeholder Matrix

This was the extent of the first version of the stakeholder management tool. Following some constructive criticisms, the model was expanded, by including more detail, and analysing the stakeholders further.

Given limited resources, it is practical to focus stakeholder management on those stakeholders with the most influence. It is recognised that all stakeholders are important, and their needs should be addressed. However, it is also a requirement of business to make the most out of limited resources. Thus it is more pragmatic to concentrate on those stakeholders with high influence potential on the project. With the stakeholders identified through the use of this quadrant analysis, half the quadrant diagram, with respect to the high influence stakeholders, is then subjected to further analysis.

Figure 6.4 shows an example of the stakeholder listing and determination of the roles and functions of each stakeholder. In the example, 33 stakeholders have been
identified, and their influence and interest in the area has been suggested. It is another view of the quadrant diagram, one that is more suitable to manipulation than the example shown in Figure 6.3, as it is electronic.

Some instances of use generated large numbers of influential stakeholders – in these cases stakeholders were further divided into internal and external stakeholders. This reduced the numbers of stakeholders that have to appear on graphs and made the graphs less cluttered. Internal stakeholders have different methods of management and contact to the external stakeholders and the analysis provided an insight into the extent of the relative influences of external and internal stakeholders.

<table>
<thead>
<tr>
<th>Stakeholder</th>
<th>Influence</th>
<th>Interest</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Stakeholder 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder 33</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.4: Example of a Stakeholder Listing
Influence was determined by the ability of the stakeholder to materially affect the success of a given project. Interest was determined by how much notice any given stakeholder would pay the project under consideration. The listing is a product of consensus amongst the group. The researcher made extensive use of the modified Delphi approach as discussed in Chapter Five to achieve this end result. Where the researcher was not directly involved in applying this tool, it was left up to the teams themselves as to how they achieved consensus. The Delphi Technique was demonstrated to various team members, and the researcher offered to help apply the technique in the various teams.

The next stage is to develop stakeholder roles and functions. The differentiation between roles and functions were as follows: a stakeholder is defined by its role, whereas a function is defined and determined by the stakeholder.

Roles and functions are terms that would be defined by a study team, and would be relevant to the particular project that required stakeholder management. An example of defined roles and functions is detailed below, taken from the work performed by the strategy team (see section 6.4.2). The identified roles were:

- Provider – the stakeholder provided goods to Rolls-Royce
- Acquirer – the stakeholder purchased goods from Rolls-Royce
- Authorise/accept – the stakeholder authorised or accepted various actions by Rolls-Royce
- Define requirements – the stakeholder defined binding requirements to which Rolls-Royce had to operate

The identified functions were:

- Programme – the stakeholder influenced the in-service use of the product
- Procurement – the stakeholder could influence procurement activities
- Support - the stakeholder could influence the support of the RR product
- Design – the stakeholder could influence the design of the RR product

Roles were independent of the stakeholder, whereas functions were dependent upon the stakeholders. The roles and functions were not defined to be generic; they were defined to suit the specific project. However, the principles of definition are generic.

Once the roles and functions had been defined, the stakeholders were then apportioned according to their role or function. This exercise clearly illustrates where and why stakeholders were highly influential. Figure 6.5 shows an example of the graphical summary of splitting the internal stakeholders according to their function(s).

![Internal Stakeholders by Function](image)

**Figure 6.5: Example of a Summary Graph – Internal Stakeholders by Function**

The further analysis can also be done with the production of various summary pie charts, counting the number of stakeholders with a given number of functions and/or roles. An example of this type of graphical summary is given in Figure 6.6. It is helpful for those doing the analysis to use a High/Medium/Low rating, where Low means having one role or function, Medium possessing two roles or functions, and High having three or more roles or functions.
The stakeholder analysis was very valuable. It showed exactly which stakeholder(s) were critical to project success, and why they were critical. Identification of external stakeholders allowed for mitigation of their influence to be planned into the project at an early stage. The analysis revealed that there were many stakeholders that combined several influential roles.

The graphical output produced is very valuable. It shows at a glance where the balance of influence lies. Grouping of stakeholders by the number of roles or functions initially caused some confusion, as stakeholders with only one role or function were sometimes perceived as not being influential. However, this process did pinpoint those stakeholders that were influential across the business, whereas others' influence was very focussed to a single area.

This type of analysis is extremely useful in the management of stakeholders. It tells the team leader exactly why stakeholders are influential, and what their interests are likely to be. Consequently, tailored reports can be issued to stakeholders, eliminating the information that is not of any interest/importance to them, and concentrates on what they need to know. It was noticed that some of the stakeholders were extremely influential in one area, but had no interest or influence in other areas. Identifying high

Figure 6.6: Stakeholder Summary – Functional Stakeholders
magnitude stakeholders (those with high influence in more than two roles or functions) does not mean that stakeholders with influence on only one role or function can be ignored. Where it does help is in the identification of the influence and in the pinpointing of what will be of interest to that particular stakeholder.

Stakeholder management is not a one-off exercise. It has to be continually revisited in order to assure both the working party and the stakeholders that everyone’s views are being addressed. Frequent reports were issued by the project management office, tailoring certain reports to the requirements of each stakeholder, as developed in the brainstorming session. The full report was also made available to the stakeholders, for information purposes and record keeping. An iterative approach, continually reviewing the stakeholders, ensures that stakeholder requirements are uppermost in the minds of any working party. The iterative nature of the process tries to avoid the effect of time on the output by forcing re-evaluation at different time periods. This produces a common set of answers. It is acknowledged that there will be changes to the answers dependent upon the history and maturation, but the changes to the answers may not be of great importance. If the changes are, then the iterative process will ensure that the changes are kept.

The tool was further validated through corporate acceptance. It has since been used by several other teams in Rolls-Royce. It has been recommended to other teams, some of which are wholly unconnected with the IT environment the tool was originally designed for. However, all reports have been favourable, to the extent that the tool is currently being considered for incorporation into the Rolls-Royce Quality Management system.
### Table 6.1: Summary of the Stakeholder Management Tool

<table>
<thead>
<tr>
<th>Name:</th>
<th>Stakeholder Management Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool for monitoring portfolio change</td>
</tr>
<tr>
<td>Purpose:</td>
<td>• To analyse what makes certain stakeholders important to the success of a project, and why they are important</td>
</tr>
<tr>
<td>Reason for Development:</td>
<td>• Stakeholder management central to project success</td>
</tr>
<tr>
<td></td>
<td>• Most stakeholders uninterested in most of the project, only wanting to keep an eye on the bit that concerns them. Sending them everything to do with the project wastes their time</td>
</tr>
<tr>
<td></td>
<td>• Needed a tool to define who to concentrate on, and what they were interested in, in order to achieve buy-in</td>
</tr>
<tr>
<td>Format:</td>
<td>Graphical</td>
</tr>
<tr>
<td>Requirements:</td>
<td>Simple analysis, spreadsheet, various business experts and a meeting room</td>
</tr>
<tr>
<td>Strengths:</td>
<td>• Clearly identifies who is important, where, and why</td>
</tr>
<tr>
<td></td>
<td>• Focuses team on stakeholders at an early stage</td>
</tr>
<tr>
<td></td>
<td>• Reduction of information sent to stakeholders, giving them only what they are interested in, gives better relations with stakeholders</td>
</tr>
<tr>
<td>Weaknesses:</td>
<td>• Requires effort by the team to create</td>
</tr>
<tr>
<td></td>
<td>• Requires a lot of information censoring when sending out targeted reports</td>
</tr>
<tr>
<td></td>
<td>• Provides only information, not decisions</td>
</tr>
<tr>
<td>Suitable for Generic</td>
<td>Yes</td>
</tr>
<tr>
<td>Application?:</td>
<td></td>
</tr>
</tbody>
</table>

### 6.6 Basic Entity Relationship Analysis

During the course of the final phase of study, it was apparent that some form of analysis would be required to find out what was requiring data, and how that data is to be used. (This is shown on Figure 6.1 as ID 3.4 and ID 3.5). This methodology is used as part of the SSADM. Within the SSADM, it is termed logical data modelling [Downs et al, 1992], producing a logical data structure.
The purposes of an entity relationship model are (from [Downs et al, 1992]):

1. To identify and define the data which the organisation needs in order to operate
2. To describe this data in a logical, abstract manner, independent of a physical implementation, in such a way to assist further design
3. To facilitate communication

Traditional use of logical data modelling is with respect to a specific IT system. In the case study, the domain is enlarged to reflect an entire system, not just concerned with IT.

An entity relationship model can embody a large volume of data. This is valuable as a cross-check and validation for output of other modelling techniques, such as the architecture analysis tool (Section 6.9), the stakeholder analysis tool (Section 6.6), and the contextual data flow diagrams (Section 6.7).
Figure 6.7: Example of Simple Entity Relationship Diagram
Figure 6.7 contains various symbols that were used as aide memoirs during the creation process. Black diamonds were used to suggest possible keys. Black crosses were used to suggest data from different databases. Black squares represented data types that were not currently collected. These were ideas that occurred during the creation process, and needed to be captured. It is not suggested that the symbolism be ruthlessly adopted. Those specific symbols were used because it was available on the drawing package at the time.

The basic principles of entity relationship modelling are simple. There are entities, relationships, and attributes. Entities are “things that are of interest to the organisation” [Downs et al, 1992]. The definition is purposely ambiguous, as there will be many things that will be of interest to a company and bounding the definition will reduce the flexibility of the methodology.

A relationship will exist between entities. A relationship can have a name and degree. The degree defines the specific relationship between the entities – one-to-one, one-to-many and many-to-many. A one-to-one relationship implies that one instantiation of an entity will have a named relationship with one instantiation of another entity.

Many-to-many (or M:N) relationships are viewed with suspicion by data modellers [Downs et al, 1992]. M:N relationships often conceal entities important to in information system, and about which data needs to be captured.

An additional refinement to relationships, optionality, can be introduced to indicate that an entity may have an optional or mandatory participation in a relationship. As part of the optionality, exclusivity can be invoked, showing that membership by an instance of an entity type of one relationship prohibits membership by that entity instance of the other relationships (Reilly provides a more detailed demonstration of using the ERD approach in Thayer and Dorfman [1997]).

Figure 6.7 illustrates the use to which basic entity relationship modelling can be used within the context of IT strategy development. As such, the optionality and exclusivity refinements have not yet been introduced. Entity relationship modelling
can be used to provide an overview model of a business system and is not intended to describe a single information system. It is meant to capture details about entities and data requirements and to provide a validation to other tools.

Both the traditional process and the modified process of compiling and ERD is iterative. The initial starting point is the creation of a study team to focus on producing the diagram. This group should be small, to enable frequent meetings, and contain significant domain knowledge. The size of ‘small’ is very much context sensitive, and would depend on the nature of the company. It should be large enough to have a significant degree of business knowledge, and small enough that its members can meet frequently. Due to the simplified nature of the entity relationship diagramming methodology in use, previous experience of software development is not necessary.

Figure 6.7 is based on an entity relationship diagram approximately halfway through the iterative process of development. The rules about relationships remain the same. Keys are sometimes defined, and data types are identified as belonging to various entities. The entity relationship diagram went through many iterations, incorporating feedback. It was not felt necessary, unlike in a formal ERD, to define all the keys and data types, as the model was not going to be used to structure a database. It was enough to identify a need for some type of datum, rather than a specific data type. This presented an opportunity to identify needs without getting into low level data analysis that was unnecessary.

It was also not felt necessary to identify keys for each table. Again, this was because the structure did not have to be developed in order to facilitate retrievals of data. The main aim of the process is to get the non-IT professionals to examine a business system from a different viewpoint. It is also possible to add short hand icons to the diagram.

A further modification to the ERD is the use of shorthand labels to represent certain characteristics. For example, it may be of use to identify possible keys, or data that is not currently available to the business. Figure 6.7 illustrates the ability of marking up the diagram, by putting various icons next to data types or potential keys. The
shorthand icons do not necessarily have to be included in the final diagram, but they could well be useful to the development team as aide memoirs.

By changing the use and depiction of the ERD means that the nature of the ERD itself is changed. It does not have to be rigidly structured, as it will not be used to develop a structured database. Instead, it is used to structure knowledge about a business system, identifying issues, which can then be developed further, as necessary. This is a novel use of the ERD methodology. The researcher is unaware of anyone else using an ERD in this fashion.

Table 6.2: Basic Entity Relationship Analysis Summary Table

<table>
<thead>
<tr>
<th>Name:</th>
<th>Basic Entity Relationship analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Tool for specifying portfolio change</td>
</tr>
<tr>
<td>Purpose:</td>
<td>To simply model the structure of the data requirements from an holistic system viewpoint</td>
</tr>
<tr>
<td>Reason for Development:</td>
<td>Needed to be able to define how the business system would work</td>
</tr>
<tr>
<td></td>
<td>Needed</td>
</tr>
<tr>
<td>Format:</td>
<td>Pictoral</td>
</tr>
<tr>
<td>Requirements:</td>
<td>Simple analysis, PowerPoint, various business experts and a meeting room</td>
</tr>
<tr>
<td>Strengths:</td>
<td>Summarises a very complex position on single page</td>
</tr>
<tr>
<td></td>
<td>Standard, well documented approach</td>
</tr>
<tr>
<td>Weaknesses:</td>
<td>Complex situations give rise to complex pictures</td>
</tr>
<tr>
<td></td>
<td>Very complex pictures require a special tool to draw the diagram</td>
</tr>
<tr>
<td></td>
<td>Subtleties of diagrammatic representation can be lost if the team is not skilled in the derivation and use of entity relationship analysis</td>
</tr>
<tr>
<td>Suitable for Generic Application?:</td>
<td>Yes</td>
</tr>
</tbody>
</table>
6.7 Contextual Data Modelling Tool

In a similar fashion to the ERD, contextual data modelling is a well-known methodology. However, in this research it is used in a novel fashion.

This tool is a modification of a tradition data modelling tool. A data flow diagram shows the interaction of data within the proposed system. It is a visual representation of how elements within the system interact both with elements internal to the system and external. Data flow diagrams give a graphic representation of entities, processes, data flows and stores, and how they relate. Contextual data flow modelling features both in the UML process and the SSADM (discussed in Appendix Two and section 2.7.2 respectively).

In a very real sense, this tool looks at the same things as an entity relationship modelling exercise, but from a different viewpoint. This makes it valuable for providing a cross-validation mechanism for the entity relationship diagram.

In a departure from the original form of contextual data flow diagrams, the rules are relaxed. The principle reason for this is that the tool is designed to be used by non-IT professionals. The tool is to get the focus group thinking about how the data interacts, giving a different view to the entity relationship diagrams. This presents an opportunity to clarify their thoughts regarding the situation, and how they are portraying it.

A fundamental part of this tool is the creation of data input and output tables. These link directly to the entities in the entity relationship diagrams, as they monitor what data is required for each process, and what data that process will produce.

The process of analysis is task-based. The analysis begins by defining generic tasks associated with the stated goal. The tasks are derived from the analysis of future business requirements using the data gathered from the implementation of the feature analysis methodology discussed in Chapter Five. Once the generic tasks are defined, then the data that those tasks would need can be developed.
To develop the data flows, the modified Delphi technique was employed by the researcher on a group of business representatives. An iterative approach was used, so that refinements could be added. Once an initial data flow diagram was created, it was subjected to various scenarios to test it in an iterative fashion. The modifications could then be applied accordingly. The use of the Delphi technique is not an integrated part of the toolset – it is merely a good way to achieve consensus on issues. The teams in which the researcher was not a direct participant were not obliged to use the Delphi technique, but were aware that the output of the team had to represent a consensus of opinion.

Figure 6.8 shows an example of the contextual data flow diagram. In this particular case, the process is task focussed, and the system is directed towards fulfilling that particular task. The task depends upon several functions and their management. Documentation supporting some of the functions can be shown, along with requirements that are outside of scope of the study, procedure authorisation gates and co-ordination events.
Figure 6.8: Example of Contextual Dataflow Diagram
Table 6.3: Contextual Data Flow Diagram Summary Table

<table>
<thead>
<tr>
<th>Name</th>
<th>Contextual data flow diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Tool for specifying portfolio change</td>
</tr>
<tr>
<td>Purpose:</td>
<td>• To identify data flows from a business viewpoint</td>
</tr>
<tr>
<td></td>
<td>• To cross check an ERD</td>
</tr>
<tr>
<td>Reason for Development:</td>
<td>• To identify data flows in a product data management process</td>
</tr>
<tr>
<td>Format:</td>
<td>Pictorial</td>
</tr>
<tr>
<td>Requirements:</td>
<td>Business knowledge, Microsoft PowerPoint or equivalent</td>
</tr>
<tr>
<td>Strengths:</td>
<td>• Provides a good indication of data complexity, if present</td>
</tr>
<tr>
<td></td>
<td>• Cross checks the ERD – data types/requirements can be discovered by tracing their movement through the business</td>
</tr>
<tr>
<td>Weaknesses:</td>
<td>• Is only as good as the business knowledge extracted</td>
</tr>
<tr>
<td></td>
<td>• Is not as detailed as a traditional data flow mapping exercise</td>
</tr>
<tr>
<td>Suitable for Generic Application?:</td>
<td>Yes</td>
</tr>
</tbody>
</table>

6.8 “Use or Lose” Test

This test was not in the planned tool set. However it was found to be of great value in determining the importance of systems. Contrasting with the tool above, this tool is for use with the second strategy, that of modification. It was employed only by the strategy development team.

The modification strategy required a good look at the existing structure to see if there was any duplication of resources. A rationalisation exercise was initiated to examine whether all of the existing systems were actually required. Use of several of the tools described in Section Two, particularly feature analysis, indicated that several systems were performing apparently similar functions.
Each of the 30 systems in the review by the strategy development team was subjected to a “use or lose” test. This test took the form of sequential reasoning flow, as shown in Figure 6.9:

As can be seen from Figure 6.9, this test embodies a significant amount of information collected from other tools. It required detailed knowledge of the current system capabilities. This knowledge has been captured by the feature analysis, and in some circumstances from quality documentation. The second stage also requires knowledge gained from feature analysis, but also makes extensive use of expert judgement to ascertain whether modernisation is possible. The third stage ascertains whether the modernisation is feasible, given cost and time constraints.

The ‘Use or Lose’ process was found to be an extremely useful summary tool to encapsulate the entire SSM process up to Stage Three (Strategy Development) by the strategy development team. By adhering to this framework, it was possible to build cases to build to keep or dispose of various systems, each case detailing why the system was to be kept or disposed. It proved very useful in the presentation to management, as it allowed clear, concise answers to be given, in line with how the management would view the problem of IT systems.
Table 6.4: ‘Use or Lose’ Test Summary Table

<table>
<thead>
<tr>
<th>Name:</th>
<th>‘Use or Lose’ test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Tool for understanding existing portfolio and identifying portfolio change options</td>
</tr>
<tr>
<td>Purpose:</td>
<td>• Quick method to determine the value of a system by its use</td>
</tr>
<tr>
<td>Reason for Development:</td>
<td>• Discovery of many very small systems in Rolls-Royce, often used by only a small number of people</td>
</tr>
<tr>
<td></td>
<td>• Needed to find out whether these should be kept, discarded, or amalgamated</td>
</tr>
<tr>
<td>Format:</td>
<td>Text</td>
</tr>
<tr>
<td>Requirements:</td>
<td>Users, meeting room, facilitator</td>
</tr>
<tr>
<td>Strengths:</td>
<td>• Simple and quick</td>
</tr>
<tr>
<td>Weaknesses:</td>
<td>• Not very detailed</td>
</tr>
<tr>
<td></td>
<td>• Needs extensive user input</td>
</tr>
<tr>
<td>Suitable for Generic Application?:</td>
<td>Yes</td>
</tr>
</tbody>
</table>

6.9 Quick Architecture Analysis Technique

This tool was developed to provide detail for the proposed replacement strategy. It was developed as an aid in illustrating the scale and complexity of proposed migrations to a new infrastructure. The tool is simple – the proposed architecture is graphically represented in a very simple, ‘bubble’ form, together with proposed connections (if any). Using this methodology, an ERP system that required part data from a PDM system would be connected. Once this is done, the existing architecture is mapped on to it, from a data and capability point of view. For example, are some of the functions in an existing system replicated in any of the new systems? If so, the name of the existing system is put in the new system bubble. This is repeated for each system in the existing architecture. If an existing system has functionality that is not replicated under the new architecture, it is put outside the bubbles. Figure 6.10 shows an example of the ‘bubble’ picture.
Figure 6.10: Example of the Quick Architecture Analysis Technique
It is simple technique, with a simple output. However, the process of creating the output requires detailed functional knowledge of both the proposed architecture, and the current IT infrastructure.

It was applied by both the strategy team and the future business capability study group to the specific context of the Naval Nuclear business of Rolls-Royce. In both cases it proved to be exceedingly useful. The difference between the two applications was a matter of scale. The strategy team employed it for PDM systems only. The future business capability study group used it to portray the impact of migration on all the databases concerned with the business.

The knowledge of the new architecture was easily gained, although specific functionality was occasionally unclear. This was remedied by referral to corporate experts in their respective fields. This was usually because the systems were in development at the time, which meant that capability was constantly changing.

Current business IT capability was described in the quality literature that Rolls-Royce produced as part of its ISO 9000 certification. It was necessary to present information regarding current and proposed IT architecture at various executive Board meetings. It was necessary that these meetings be non-confrontational, as the intent was to establish whether the proposed architecture captured everything that it needed to. The strategy development team performed the analysis on an existing suite of approximately 30 databases. The proposed architecture was encapsulated as four principle databases. The existing system covered various aspects of product data.

A diagram was produced similar to that in Figure 6.10. This was emailed to key business personnel, and feedback incorporated. This occasioned some heated debate when capability was developing, but consensus was eventually reached. The Delphi technique of iteration and analysis proved very useful in this respect. As before, the Delphi technique is not mandated for use. It is just an effective means of gaining consensus. All the teams were free to choose whether they wanted to use the technique or not.
This technique is very simple, and its main strength is the ability to portray a very complex situation in one picture – "a picture is worth a thousand words." Once the background knowledge was assembled, the analysis was very quick to perform. Both teams produced the analysis within one day. It was certainly exceedingly useful in explaining the scale of a proposed migration to senior management. It was also useful as an aid to work out, at a high level, whether important functionality was being left out of the proposed architecture. This conclusion can also be drawn from the feature analysis, and so the use of both techniques provides a significant amount of corroboration.

The technique does have two significant weaknesses. Firstly, it is not a decision tool. It summarises information; it does not give any guidance as to what to do once you have summarised it. The decision is left to the decision-makers. The tool is intended to simplify the situation to make details apparent. Secondly, it requires a large amount of background knowledge about the proposed architecture and the current architecture. If this knowledge is not available, the process of compiling it could be lengthy.
### Table 6.5: Quick Architecture and Systems Analysis Summary Table

<table>
<thead>
<tr>
<th>Name:</th>
<th>Quick Architecture/Systems Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Tool for understanding the existing portfolio and identifying possible portfolio change options</td>
</tr>
</tbody>
</table>
| Purpose:         | • Quick method to show the changes required to the existing architecture  
                   • Systems analysis uses the same principles, but for a given function |
| Reason for Development: | • Need to show senior managers the scale of change to the architecture  
                            • Need to show the capabilities of the existing systems compared to the proposed systems |
| Format:          | Pictorial |
| Requirements:    | Users, meeting room, facilitator |
| Strengths:       | • Simple and quick  
                   • Very powerful descriptive tool  
                   • Excellent for illustrating the scale of change in an intuitive fashion |
| Weaknesses:      | • Not very detailed  
                   • Needs a considerable amount of system knowledge to develop |
| Suitable for Generic Application?: | Yes |
6.10 Summary

The third and final phase of study fulfilled the aim and objectives of the thesis, by developing and using a toolset for the proactive management of the Rolls-Royce IT portfolio. This phase of the research utilised the most tools, and was the final iteration of the development of these tools during the course of this research. The use of the heavily modified SSM-based methodology was well received; business representatives liking the flexibility to be able to adapt tools to their use whilst simultaneously have a structure to hang ideas and concepts on.

It should be noted that none of the tools or methodologies contain decision rules – they do not supplant managers. What they do provide is detailed information, conveniently displayed. The simplicity of many of the tools is a tremendous strength – senior management can easily grasp what the problem is, and make decisions using superior information in a very short space of time. Being able to divide large projects into smaller projects, with faster payback periods was very advantageous, and also minimised the impact that the changes had by spreading them out.

Many of these tools in this phase of research were newly developed since the pilot stage. This was because they were developed in response to problems encountered during the pilot stage. The iterative nature of the process is ideal for capturing these issues, and developing a tool to resolve them.

The study ended with the completion of many of these projects. The next chapter summarises the results and experience gained from the use of this toolset within Rolls-Royce Naval Marine.
References:

Chapter 7: Business Context & Results

7.1 Introduction

This chapter documents the experience from using the various tools and methodologies that has been gained during the three periods of research. The aim and objectives are fulfilled, by having a suite of tools and methodologies available for use in proactive IT portfolio management. It describes the business context, and how environmental factors can influence the IT portfolio management strategy chosen and implemented. It then evaluates the toolset in each period of study, showing how useful each tool or methodology was. In doing this, it documents the experience of the many teams that were described in the last chapter. The many tools and methodologies employed were developed through an iterative process, and therefore have a particular, Rolls-Royce relevance. However, the researcher believes that many of the tools and methodologies can be applied in other industries, as they are very generic and do not contain decision rules, and are merely aids in presenting complex information in a simple fashion.

7.2 Business Context of the Third Phase of Study

The business context of the third phase of study is very important. At that specific time, costs of IT projects became highlighted, coupled with a lack of identifiable returns from system implementation. A global recession, complicated by the terrorist event of 9th September 2001, now known as ‘9/11,’ and the prospect of war in Afghanistan, meant that global markets were very depressed. Aerospace engineering in particular was hit hard, with reductions in orders and spares markets being drawn downwards due to cannibalisation of inactive aircraft.

These reasons meant that Rolls-Royce was paying close attention to costs, seeking effective and cost-efficient means to bring down costs and preserve thin profit margins. One consequence of this was a requirement of realisable returns from IT investment. Any investment must be seen to have realisable gains, preferably in the short term to gain support.
One aspect Rolls-Royce management were examining was product data management. New market conditions obliged a much more focussed and immediate response. This attitude was extended to other areas of the IT infrastructure, when the approach used by the researcher for product data management showed results.

7.3 Using the Portfolio Management Framework.

As has been shown, a significant volume of work had already been performed on the subject of product data management in the Nuclear business site of Rolls-Royce. The PDM team was to bring this all together, and subject it to a more detailed examination, in light of new requirements and conditions. This meant that the framework proposed earlier was not followed by the PDM team as closely as the other teams were able to, as the other teams had the benefit of being able to apply a fuller toolset from first principles. As a result, the PDM team findings are presented separately to those of the other four teams.

7.3.1 PDM Team: Stage One – Problem Definition

The problem was dictated by the terms of reference of the group. However, the group still needed to discover the cause and the ramifications. Business analysis skills were paramount at this stage, as cash flows had to be analysed, to work out where money was being spent, how it was being spent, and highlight areas of possible overspend or haemorrhage.

Questionnaires had already been sent out (see section 7.9), these were used to show what users perceived the problem to be. A structured review of contractual requirements was conducted, in order to determine contractual commitments. A review of standards was also conducted at the same time, looking at what standards were applicable, and what requirements those placed on the business. Combined with stated commitments regarding future business from Rolls-Royce, this provided a good picture of what was desired – the non-problem state.

There was also what was termed the ‘flip side’ – what was wrong with the current situation? Again the questionnaires provided data. The previous modified feature analysis (see section 6.5) conducted on the PIMS system was brought in (IDs 3.1. 3.2
and 3.3 in Figure 6.1), firstly to show what users desired, and secondly to show where PIMS was not meeting expectations. The primary view used here was the feature list itself, and the system comparison view of PIMS alone. This evaluated PIMS against the feature list, rather than the original intent, which is to compare system against system.

Graphical defect analysis (see section 7.8) was also used, to show the volume of defects, and their severity. The hierarchical defect analysis was also used, to show the number of serious defects, the number of defect due to business change, and the number of minor, code-related defects. The intention here was to show that the current business requirement had been stable for many years, with the expectation, on reading back to the corporate strategy, that the business requirements would be unlikely to change, although some new ones would be required if the business were to be expanded rather than grown.

Scare stories (see section 4.4 and Appendix Two) were reviewed, looking for similar situations, and similar circumstances. They were also used in order to steer away from taking on too many problems, and the dangers of widening the scope.

The output from this stage was a memo to the relevant Heads of Department, detailing the problem, its causes, and its ramifications. It used a similar style to that shown in section 6.3.
Table 7.1: PDM Team Tool Use in Stage One

<table>
<thead>
<tr>
<th>Tool Used</th>
<th>Reason</th>
<th>Usefulness</th>
<th>Relevant Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contract analysis</td>
<td>Defines contract requirements unequivocally</td>
<td>HIGH</td>
<td>-</td>
</tr>
<tr>
<td>Standards analysis</td>
<td>Defines requirements from standards unequivocally</td>
<td>HIGH</td>
<td>-</td>
</tr>
<tr>
<td>Business analysis</td>
<td>Examines cash flows relating to the subject, looking for anomalies or high flow</td>
<td>HIGH</td>
<td>5.3</td>
</tr>
<tr>
<td>Graphical Defect Analysis</td>
<td>Picture of defect patterns relating to the PIMS system</td>
<td>Low</td>
<td>5.8</td>
</tr>
<tr>
<td>Hierarchical Defect Analysis</td>
<td>Intuitively show the stability of business requirements, and the stability of the system itself</td>
<td>HIGH</td>
<td>5.7</td>
</tr>
<tr>
<td>Feature analysis – feature list</td>
<td>Shows the \textit{status quo post}, allowing comparison with the \textit{status quo ante}</td>
<td>HIGH</td>
<td>5.5</td>
</tr>
<tr>
<td>Feature Analysis – system view</td>
<td>Pinpoints areas of user dissatisfaction with the current system</td>
<td>HIGH</td>
<td>5.5.1</td>
</tr>
<tr>
<td>Questionnaires</td>
<td>Provides snapshot of user opinion</td>
<td>Medium</td>
<td>5.10</td>
</tr>
<tr>
<td>Rich Pictures</td>
<td>Provide an intuitive description of the situation</td>
<td>HIGH</td>
<td>5.6</td>
</tr>
<tr>
<td>Story telling</td>
<td>Provides some guidance as to where causes of problems might lie</td>
<td>Medium</td>
<td>4.4</td>
</tr>
</tbody>
</table>

None of the tools required modification beyond that already discussed. Statistical analysis of failure data had been performed earlier, but the results were not deemed useful and were not included in the PDM team data use. The researcher feels that this is an important point, reflecting the use of statistics in general, and the particular use...
of statistics in this case. The researcher would not recommend the use of the particular statistical manipulation performed within this context.

The business analysis, including contract and standards analysis used no set methodology. However, they are vital to determining the ramifications of the problem, and in some circumstances, can form part of the problem themselves.

The output from this stage was a small report to the team champion and interested parties, detailing the problem, its causes, and ramifications. This memo was circulated for comment, and the champion secured funding for the next stage, which was option development. The modified Delphi technique was particularly useful, as it ensured that all members of the team were completely bought into the problem definition.

The graphical defect analysis was not very useful. It was used because it had been done and provided supporting evidence. However, it was observed by the researcher that the output was not much affected by the inclusion of the graphical defect analysis and the decision of the senior management to release funding for the option development was probably more influenced by the business and contract analysis.

7.3.3 PDM Team: Stage Two – Option Development

Option development was for the PDM team was a short but intense period, lasting two weeks, full time. The team had the problem definition, the cause(s) and the ramifications. The next step was to work out possible ways in which these problems could be mitigated. Many tools from the tool set were used, and it is from this point that the framework was followed, and the tools used from scratch. The exception to this was the feature analysis, and stakeholder analysis.
### 7.3.4 Results of PDM Team Tool Use: Stage Two

#### Table 7.2: PDM Team Tool Use in Stage Two

<table>
<thead>
<tr>
<th>Tool Used</th>
<th>Reason</th>
<th>Usefulness</th>
<th>Relevant Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified Delphi Approach</td>
<td>Achieves consensus of opinion from teams/experts</td>
<td>HIGH</td>
<td>4.10</td>
</tr>
<tr>
<td>Business analysis</td>
<td>Provision of financial information</td>
<td>HIGH</td>
<td>5.3</td>
</tr>
<tr>
<td>Basic entity relationship analysis and diagram</td>
<td>Models the business system, and details relevant entities</td>
<td>HIGH</td>
<td>6.6</td>
</tr>
<tr>
<td>Knowledge base</td>
<td>Background knowledge to provide starting point for developing options</td>
<td>HIGH</td>
<td>4.4 &amp; 5.4</td>
</tr>
<tr>
<td>Story telling</td>
<td>Provides lessons learnt from other industries/technologies; doesn’t repeat history</td>
<td>Medium</td>
<td>4.4</td>
</tr>
<tr>
<td>Expert judgement</td>
<td>Provides estimation of possible costs/efforts; provides information</td>
<td>HIGH</td>
<td>4.7</td>
</tr>
<tr>
<td>Stakeholder management</td>
<td>Helps manage stakeholders efficiently and effectively</td>
<td>HIGH</td>
<td>6.5</td>
</tr>
<tr>
<td>Feature Analysis – system view</td>
<td>Compares the features of various systems/options</td>
<td>HIGH</td>
<td>5.5.1</td>
</tr>
<tr>
<td>Feature Analysis – strategic intent view</td>
<td>Compares features to strategic intent, showing a capability gap</td>
<td>Medium</td>
<td>5.5.3</td>
</tr>
<tr>
<td>Feature Analysis – process view</td>
<td>Links features to the relevant processes</td>
<td>Medium</td>
<td>5.5.2</td>
</tr>
<tr>
<td>Feature Analysis – cost view</td>
<td>Links features to cost to attain</td>
<td>None</td>
<td>5.5.4</td>
</tr>
<tr>
<td>Lifecycle costing</td>
<td>Provide through life costs for associated hardware</td>
<td>HIGH</td>
<td>4.6</td>
</tr>
</tbody>
</table>

This was the first trial of the stakeholder analysis, and used the first ‘incarnation’ – the quadrant diagram, without the subsequent further analysis. The tool was still useful, but the experience of use on this team provided the feedback which saw the stakeholder analysis as described in 6.5 fully materialise.
Again, the modified Delphi technique was invaluable for ensuring that everyone was bought in and ‘on message’ when discussing the project with their respective line managers. It was also useful in maximising the use of expert judgement in the various situations that required it.

The feature analysis listing was ‘pre-compiled’ from previous work and was exceedingly important in the presentation of the options to the senior management at the end. The system comparison was used to show the various options and the management could see at a glance which options performed better. This was a very powerful tool, as it has a very simple output. However, the amount of work that goes into collecting and collating the feature list is large, and could be costly, especially for large, complex systems.

The main disappointment here was the inability to utilise the cost view of the feature analysis. The problem was mentioned before, when discussing the tool itself, and was due to the difficulty in disaggregating the costs to individual features. This is a major flaw, and stopped the cost view from being used at all.

The business analysis was the key driver to the management decision. It presented the best guess as to the financial viability of each option, its associated costs and financial benefits. Clear presentation of this data was an important part of the success of the team.

A simple form of entity relationship diagram was deployed, mapping out a possible option. The diagram mapped a business system, rather than an IT system, and was very helpful to show relevant entities to the subject area. It also impressed the senior management immensely!

The application of the knowledge base had mixed results. The background knowledge was exceedingly valuable, and helped shape most of the options. However, the flow charts did not meet with much acceptance. The team members preferred to go through each possible solution to legacy systems and say why this would help or not help the current problem.
The Rolls-Royce management attended a presentation by the PDM team, where the output from these tools was used. The decision was taken to more fully detail two options, as these represented the best way forward, but one was a radical step, so the management wished to have more information before committing Rolls-Royce irrevocably.

7.3.5 PDM Team: Stage Three – Strategy Development and Implementation

This phase was the most detailed phase, and concentrated on developing two options, with the aim of being able to present both options to the Board, with the Board choosing one. This chosen option would then be able to start immediately. This meant that detailed project plans had to be developed for both options. This is not exactly how the framework was supposed to work, as only one option should have been used at this stage, but the management wanted more information before committing Rolls-Royce down one route or the other.

One of the primary problems was the costs associated with the two options. For both options, the costs were of a high order. The management needed to be as certain as possible of the benefits before they could authorise expenditure of that level. Consequently, a great deal of effort was expended by the team on the cost benefit analysis, using data from the business analysis of the previous stage.

The output of the third stage was a published report, sent to the Board and other business representatives, detailing the two options with respect to costs, benefits and issues throughout the project lifespan. The report contained two project plans, planning the implementation of each option. The report also detailed under issues, any second generation problems that the team could see arising from the implementation, together with mitigating actions. These were also included in the risk register, and had nominated champions.

The result was a decision by the Board to implement the more radical of the two options, as the Board had sufficient confidence in the costs, benefits, and implementation plan to believe that the benefits were fully achievable within the time frame specified. The radical option included work bundling, where several streams of
work could be initiated simultaneously, rather than the standard, serial approach to project tasks. It also allowed more rapid progress, as the plan included not just IT changes, but process change as well.

### 7.3.6 Results of PDM Team: Stage Three

#### Table 7.3: PDM Team Tool Use in Stage Three

<table>
<thead>
<tr>
<th>Tool Used</th>
<th>Reason</th>
<th>Usefulness</th>
<th>Relevant Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified Delphi Approach</td>
<td>Achieves consensus of opinion from teams/experts</td>
<td>HIGH</td>
<td>4.10</td>
</tr>
<tr>
<td>Basic entity relationship analysis and diagram</td>
<td>Models the business system, and details relevant entities</td>
<td>HIGH</td>
<td>6.6</td>
</tr>
<tr>
<td>Expert judgement</td>
<td>Provides estimation of possible costs/efforts; provides information</td>
<td>HIGH</td>
<td>4.7</td>
</tr>
<tr>
<td>Stakeholder management</td>
<td>Helps manage stakeholders efficiently and effectively</td>
<td>HIGH</td>
<td>6.5</td>
</tr>
<tr>
<td>Feature Analysis – system view</td>
<td>Compares the features of various systems/options</td>
<td>HIGH</td>
<td>5.5.1</td>
</tr>
<tr>
<td>Feature Analysis – strategic intent view</td>
<td>Compares features to strategic intent, showing a capability gap</td>
<td>Medium</td>
<td>5.5.3</td>
</tr>
<tr>
<td>Cost/benefit analysis</td>
<td>Links costs to benefits over the project lifespan</td>
<td>HIGH</td>
<td>5.3</td>
</tr>
<tr>
<td>Payback</td>
<td>Simpler form of CBA</td>
<td>HIGH</td>
<td>5.3</td>
</tr>
<tr>
<td>Lifecycle costing</td>
<td>Detailed examination of maintenance costs of hardware</td>
<td>HIGH</td>
<td>5.3</td>
</tr>
</tbody>
</table>

The three most useful tools here were the cost benefit analysis, the system view of the feature analysis, and the modified Delphi technique. Stakeholder management was also a highly important part of the success of the PDM team. All of these tools provided simple, intuitive outputs that were easy to present to the senior management, and they could clearly see the reasons and assumptions underlying the business proposition.

Lifecycle costing was included, as both options required an investment in hardware. One of the differentiating decisions was the difference in the maintenance costs associated with the hardware of the two options. Lifecycle costing was very useful in
this instance, but the utility might not be so high where hardware is not part of the project cost. It is possible that lifecycle costing could be applied to the software too. However, it was not in this case.

The use of payback was highly important. Phrases like "The project is likely to pay for itself within two years" have a great impact on senior management, but only if the costs and benefits are clearly explained, and the underlying assumptions are obvious. The entity relationship diagram was very impressive. It helped clarify the thought of the PDM team on many different levels and occasions.

Implementation of the project was not at the discretion of the researcher. Responsibility for implementation was passed to an experience project manager for deployment within the business. This was a Board decision. However, the implemented project closely followed the suggested plan, including the bundling of work into discrete packages.
7.4 Portfolio Management Framework for Other Teams

One outcome of the success of the PDM team was the initiation of several other teams, as described in section 6.4. These could use the framework and associated toolset from scratch, and so they have been summarised in one section.

7.4.1 Stage One Tool Use

Table 7.4: Stage One Tool Use

<table>
<thead>
<tr>
<th>Tool</th>
<th>Strategy Development</th>
<th>Future business requirements</th>
<th>Database Rationalisation</th>
<th>Configuration Mgmt Rationalisation</th>
<th>Relevant Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contract analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standards analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.3</td>
</tr>
<tr>
<td>Hierarchical Defect Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.7</td>
</tr>
<tr>
<td>Feature analysis – feature list</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.5.1</td>
</tr>
<tr>
<td>Feature Analysis – system view</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.5.1</td>
</tr>
<tr>
<td>Rich Pictures</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.6</td>
</tr>
<tr>
<td>Questionnaires</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.10</td>
</tr>
<tr>
<td>Story telling</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.4</td>
</tr>
<tr>
<td>Graphical Defect Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.8</td>
</tr>
<tr>
<td>Statistical Defect Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.9</td>
</tr>
<tr>
<td>Modified Delphi Technique</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.10</td>
</tr>
</tbody>
</table>

Key:

- Not used
- Not very useful
- Very useful

Table 7.4 shows the tool use in Stage One of the portfolio management framework for all of the teams. The strategy development team maintained an overview of the activities of the other three. The configuration management and database rationalisation teams were both initialised by the strategy development team.
Standards and contract analysis were important tools for three of the teams. The team leader of the database rationalisation team felt that the terms of reference given to the team by the strategy development team were of sufficiently high calibre not to warrant any further investigation, so these were not used. The other teams, however, did use them. All of the teams used some form of business analysis, as distinct from CBA (Cost Benefit Analysis), to a greater or lesser extent, depending upon the needs of the project.

Three teams used hierarchical defect analysis, but for different reasons. The strategy development and future business requirements used the tool to show that the business requirements have been historically very stable, and thus are unlikely to radically change in the future. The database rationalisation team used the same tool, but examined the other end of the scale. This team looked at the code-based defects for all the current tools, looking to see the relative densities of defects in the different tools. The team also looked at model-based defects, in an effort to approximate a measure of how well designed the model was.

All of the teams compiled a feature list, but the configuration management team did not need to use the system comparison view, as it was taking the IT tools as a given from the database rationalisation team. The purpose was to ensure that both the process change team (configuration management) and the IT change team (database rationalisation) were using the same set of expectations. The modified Delphi technique was used here, across teams as well as within the teams.

Rich pictures were used by all the teams, and were highly useful for describing the current situation, and making clear where potential problems were. They were also used as an explanatory tool in presentations to the senior management, showing exactly what went on, and why the problems were as they were.

Only one team used questionnaires, the future business requirements team, and this use was not very successful at all. Returns were very low. In contrast to the PDM team, returning the questionnaire was not mandated by the senior management, so many business representatives did not cooperate fully. This was an oversight by the future business requirements team. Mandating replies to questionnaires is a very
useful way of ensuring high quality returns, and the researcher would recommend bringing the senior management on board for this type of activity, when questionnaires are useful. The database rationalisation team used analysis of the output of previous questionnaires, so the lack of use of questionnaires for that team is slightly misleading.

Story telling for two of the teams was not terribly productive, although the database rationalisation team found it extremely useful and suggested modification to involve role-playing. This was the first time this suggestion had been made, and it was not made in time to modify the toolset for this particular phase. However, the researcher does feel that this suggestion has merit, and will be including role-playing in the next implementation of the tool set.

None of the teams used graphical or statistical defect analysis. The researcher suggests that these tools, in the style in which they were presented here, are of little, if any, use and should not be used in the toolset.

In terms of timeframe, the strategy development team preceded the other three, and to a large extent, helped define the problem(s) for the other teams as part of the strategy remit. As a result, the strategy development team spent a lot more time in the first stage than the other three teams. The future business requirements team spent the next longest, taking several weeks to define and scope the problem area. The data rationalisation team was the shortest, as the problem space had been fairly well sized and scoped by the strategy development team.

The strategy development team were the primary team for changing the IT portfolio. The responsibility for managing the IT portfolio was generally devolved to operational units or the Rolls-Royce IT outsourcer. However, responsibility for generating the strategy for directing IT portfolio change remained centralised within Rolls-Royce and this is where the strategy development team had political support. This also meant that all the teams had to take careful note of user requirements, especially the future business requirements group.
The output of stage one from all teams was a statement of the problem space – what were the problems, why they were problematic, and what the ramifications were. The statements were issued in the form of an internal memo. The future business requirements team adopted a slightly different strategy to the other teams, in keeping the first stage open, as the unknown can throw up unforeseen problems.
7.4.2 Stage Two Tool Use

Table 7.5: Summary of Stage Two Tool Use

<table>
<thead>
<tr>
<th>Tool</th>
<th>Strategy Development</th>
<th>Future business requirements</th>
<th>Database Rationalisation</th>
<th>Configuration Mgmt Rationalisation</th>
<th>Relevant Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified Delphi Approach</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.10</td>
</tr>
<tr>
<td>Basic entity relationship analysis and diagram</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.6</td>
</tr>
<tr>
<td>Modified Entity Relationship Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.6</td>
</tr>
<tr>
<td>Knowledge base</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.4</td>
</tr>
<tr>
<td>Story telling</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.4</td>
</tr>
<tr>
<td>Expert judgement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.4</td>
</tr>
<tr>
<td>Stakeholder management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.5</td>
</tr>
<tr>
<td>Feature Analysis – system view</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.5.1</td>
</tr>
<tr>
<td>Feature Analysis – strategic intent view</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.5.3</td>
</tr>
<tr>
<td>Feature Analysis – process view</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.5.2</td>
</tr>
<tr>
<td>Feature Analysis – cost view</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.5.4</td>
</tr>
<tr>
<td>Contextual data flow diagrams</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.7</td>
</tr>
<tr>
<td>Lifecycle Costing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.6</td>
</tr>
<tr>
<td>Payback</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.3</td>
</tr>
</tbody>
</table>

Key:
- Not used
- Not very useful
- Very useful

Stage Two sees a large number of tools being used. This is primarily because of the different natures of the remits of each team. The toolset shows that it is very flexible, and only tools that are going to be of use need be used.
Again, all of the teams used the modified Delphi technique. This has several advantages. Firstly, it ensures that all of the team members are in agreement with anything that is issued in the team’s name. Secondly, it provides a very good source of minutes for all of the meetings. Lastly, it provides an excellent audit trail to see why various decisions were made, and who was involved in making them.

Entity relationship diagrams (ERDs) were used by only the strategy development and future business requirements teams. Of these teams, only the future business requirements team used the full capability of the entity relationship diagram. It was discussed by the strategy development group whether to take forward the ERD, but it was decided to let the future business requirements team take the lead in that, and fully define the business system. Defining the basic ERD took the strategy development team approximately four afternoons, and this was later handed over to the future business requirements team for further development.

The knowledge base was again of mixed use. The knowledge itself was extremely useful, especially for the database rationalisation team and the strategy development team. The flowcharts were not particularly useful. The flowcharts were useful as a method of organising the knowledge, and were used as a tick box approach, but it was the knowledge itself that was valuable, not the manner it which it was presented. As a result, the researcher feels that the flow charts are not as useful as he expected. The teams did not have any suggestions to make on improvement however.

The stakeholder management tool was exceedingly well received. All of the teams used it. The strategy development and future business requirements teams used all of the graphs. The other two teams found it more suitable to stop at the bar chart level, and not aggregate the numbers of stakeholders with one, two, three or more roles. This is possibly because the first two teams had far more stakeholders than the other two teams, meaning that the bar chart presentation was very crowded for the strategy development and future business requirements teams. The tool has been recommended for inclusion in the Rolls-Royce Quality Management System, and has been successfully used on a number of other projects since.
Feature analysis is another tool that was very useful. The system view in particular, was highly effective in summarising a complex position, and then being able to let senior management see what that position is, with a minimum of explanation. The process and strategic intent views were not as useful, although their compilation did give the teams an exceedingly good grasp of the strategic intent of Rolls-Royce, and the Rolls-Royce processes. The researcher has since found other methods of getting to grips with processes, such as process mapping on a wall, but the feature analysis approach has the advantage that it is integrated with other views, and ties things together very nicely. The cost view of feature analysis is not useful at all. The concept suggests it should be very useful, but practical difficulties render it inoperable, and therefore not practical to include in the toolset.

Contextual data flow diagrams, and related data input and output tables, were used by two of the groups. The strategy development team used an exceedingly basic version as a check for the ERD. The future business requirements team took the concept further, and developed the contextual data flow diagram in conjunction with the far more detailed ERD. The future business requirements team spent many days developing the contextual data flow diagrams, defining terms, and cross referencing with the ERD. An iterative approach was employed throughout, with the researcher acting as facilitator and recorder. Version control was essential, and therefore all changes went through the facilitator, ensuring that all updates were correctly captured.

The contextual data flow diagram was an excellent means of extracting knowledge from people, and capturing it on paper. It has been extensively presented to various members of Rolls-Royce management, but the feedback has not been as positive as it was for the system-view of feature analysis. As a means of capturing knowledge that was previously contained within people's heads, it was very useful. The diagrammatic approach was initially greeted with great scepticism by many of the team members, especially the IT representative, and the business experts. However, all of them have become enthusiastic supporters of its use, as the benefits have been clear to them following the success of the future business requirements presentations to the Board. One of the team members is now using them as part of his succession planning pack, in which he is documenting knowledge for his successor, following his retirement in a few months' time. The diagrams were also used to validate the ERD,
as they are both looking at the same problem, just from different angles. The contextual data flow diagram would be useful without the ERD, but it becomes essential if the ERD is being used.

Hardware formed a significant aspect of three of the team’s remits. Consequently lifecycle costing was used. The feedback was largely positive, although the strategy development team found the tool slightly too involved for their needs. Having said that, the output was able to be used by other teams, and therefore was not a waste of effort.

The use of payback is interesting. The strategy development team used NPV for preference, but found the senior management to be more comfortable with payback as an initial measure of project suitability. Consequently the other teams used payback in preference. NPV wasn’t discarded, but payback was given a higher impact factor, and was mentioned more often in meetings with the senior management.

Overall, the toolset worked well. It was flexible enough that the tools could be adapted for use in different projects, and rigorous enough that the outputs found favour with the senior management and implementation permission was given in all cases.
7.4.3 Stage Three Summary

Table 7.6: Stage Three Tool Use Summary Table

<table>
<thead>
<tr>
<th>Tool</th>
<th>Strategy Development</th>
<th>Future business requirements</th>
<th>Database Rationalisation</th>
<th>Configuration Mgmt Rationalisation</th>
<th>Relevant Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified Delphi Approach</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.10</td>
</tr>
<tr>
<td>Modified Entity Relationship Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.6</td>
</tr>
<tr>
<td>Expert judgement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.7</td>
</tr>
<tr>
<td>Stakeholder management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.5</td>
</tr>
<tr>
<td>Feature Analysis – system view</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.5.1</td>
</tr>
<tr>
<td>Feature Analysis – strategic intent view</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.5.3</td>
</tr>
<tr>
<td>Contextual data flow diagrams</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.7</td>
</tr>
<tr>
<td>'Use or Lose' test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.8</td>
</tr>
<tr>
<td>Quick Architecture Analysis tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.9</td>
</tr>
<tr>
<td>Quick Systems Analysis tool</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6.9</td>
</tr>
<tr>
<td>Lifecycle costing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.6</td>
</tr>
<tr>
<td>Cost Benefit Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Appendix 5.3</td>
</tr>
<tr>
<td>Payback</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Appendix 5.3</td>
</tr>
<tr>
<td>Net Present Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Appendix 5.3</td>
</tr>
</tbody>
</table>

Key:
- Not used
- Not very useful
- Very useful

Stage three is the strategy development. It is the in-depth review and analysis of the chosen solution, working up a detailed project plan, with milestones and targets. It is an extension of Stage Two, in that many of the same tools are used, but they are used on only one of the options, the others having been discarded by the management. In
all cases, the intention is to create a multi-streamed project plan, with bundles of work being identified and linked together.

This stage is very much about achieving ‘buy-in’ by the business, and setting up a situation where the strategy can be implemented. Implementation was performed by the relevant business unit, with the strategy development team keeping an overview.

The same comments that applied for tools in Stage Two apply here, but a few more tools were used. The Quick architecture analysis tool was very useful in showing the magnitude of change, and why precautions were being taken. It was also a useful check on new system capability, ensuring that vital functions were not being left out of the new architecture. The system analysis tool was very much the same, but was used on a subject, for example tracing the links between configuration management, commodity management, and part management. The feedback from this tool was positive, especially from senior management, who found the ‘shotgun’ presentation very intuitive, and were able to see both scale and scope. They were also able to use it to find out what functions were missing from the new architecture, why, and whether they were important. This helped remove uncertainty from the decision and increased their confidence in the possible outcomes.

Cost benefit analysis was employed as a matter of course and much attention was devoted to this. The researcher feels very strongly that the concentration of all of the teams on the impact of changes to the bottom line (as illustrated by CBA) was critical to the successful use of the project recommendations.

7.5 Summary

The third phase of the research utilised the most tools, and was the final iteration of the development of these tools during the course of this research. The use of the methodology was well received; business representatives liking the flexibility to be able to adapt tools to their use whilst simultaneously have a structure to hang ideas and concepts on.
The result of the research was that the Executive Board approved extensive modernisation plans, in line with the research findings and moving away from the corporate intent. This course of action could only be taken if the Executive Board had every confidence that it could succeed. The analysis and presentation of the findings had a significant role in gaining Board approval. Consequently, the Nuclear business site had the only IT project not to have budget cuts. This project has also delivered to specifications and within budget.

By examining the process as well as the IT, process changes have made the IT, and by extension the business, more efficient. This has not only saved money for Rolls-Royce, but has enabled Rolls-Royce to extend its business into new areas, exploiting previously untapped capability. It also identified future problems early enough that they have been avoided in a cost effective and efficient manner.

The strongest lesson learned during this was that the simpler the tool, the more effective it seemed to be. The statistical defect analysis produced exactly the same result as the graphical defect analysis, but it was the graphical defect analysis that generated the most interest. That being said, neither of these two tools had a great impact, and consequently the author has classified both tools as “not very useful”. The graphical output of feature analysis made it extremely popular. There was very little interest in the many documented features, but the assessment graphs proved to be very influential in the decision-making process. The Executive Board in particular were very impressed by the graphical output, as it presented the information in a fashion that was immediately apparent. This simplicity was extended to the framework within which the analysis was conducted. The highly flexible and adaptable three-stage process performed well, and has gained many adherents within Rolls-Royce.

From the point of view of the tool set, many tools were tried, tested, modified, and some were discarded. Statistical defect analysis, in the form used here, was not a success, and the researcher would not recommend its use. In contradistinction, the Delphi-based consensus gaining approach was highly useful, and the use of this technique should not be limited to process and IT-based projects. Tables 7.6, 7.7 and 7.8 can also be taken as a summary – tools/techniques which had a favourable
response from all the teams are ones that should be used at every opportunity. If there are other flags attached, then the tools should be used with care as to the specific requirements of the situation. Table 7.7 presents a very concise summary of the researcher’s experience:

**Table 7.7: Summary of Researcher’s Experience with the Tools and Methodologies**

<table>
<thead>
<tr>
<th>Highly Useful</th>
<th>Useful</th>
<th>Not Useful</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified Delphi Approach</td>
<td>Contract analysis</td>
<td>Graphical Defect Analysis</td>
</tr>
<tr>
<td>Business analysis</td>
<td>Standards analysis</td>
<td>Statistical Defect Analysis</td>
</tr>
<tr>
<td>Feature analysis – feature list</td>
<td>Story telling</td>
<td>Feature Analysis – cost view</td>
</tr>
<tr>
<td>Rich Pictures</td>
<td>Knowledge base</td>
<td></td>
</tr>
<tr>
<td>Payback</td>
<td>Modified Entity Relationship Analysis</td>
<td></td>
</tr>
<tr>
<td>Cost Benefit Analysis</td>
<td>Feature Analysis – strategic intent view</td>
<td></td>
</tr>
<tr>
<td>Quick Systems Analysis tool</td>
<td>Contextual data flow diagrams</td>
<td></td>
</tr>
<tr>
<td>Quick Architecture Analysis tool</td>
<td>‘Use or Lose’ test</td>
<td></td>
</tr>
<tr>
<td>Expert judgement</td>
<td>Lifecycle costing</td>
<td></td>
</tr>
<tr>
<td>Stakeholder management</td>
<td>Net Present Value</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Questionnaires</td>
<td></td>
</tr>
</tbody>
</table>

Business analysis skills were of high importance and use. It is the researcher’s opinion that the business analysis skills possessed by the researcher were one of the principle reasons why the researcher was able to establish a good professional credibility in the second period of study. They are essential skills for an IT portfolio manager, as they allow the manager to understand what will have an impact on the bottom line, and how the balance sheet will be affecting by IT decisions. They will also allow the manager to gauge the impact of change on the profit and loss account as well.

One point that emerged was that the business representatives all requested more metrics. This was the reason for the inclusion of a GQM aspect into the framework. However, looking at the summary table, there are not many metric-based tools/techniques present, and those that are, are either simplistic or were not well
received. Payback is an example of the former, and lifecycle costing an example of
the latter. This discussion will be covered in more detail in the final chapter, when a
review of further work is defined.

The toolset was rapidly adopted by the many working parties, and this has given it a
very thorough testing within the Rolls-Royce Naval Marine business. Such as been
the success of the toolset, that some of the components have been put forward for
inclusion in the Rolls-Royce Quality Management System (RRQMS), which would
see the whole of Rolls-Royce using them. At the time of writing, the feature analysis
and stakeholder management have been approved for inclusion within the RRQMS.
Approval is only gained when it can be proved to a quality management board that the
tool will actively promote increased effectiveness and/or quality.

IT portfolio management is not well reported in the public domain, and the production
of framework and toolset, tested in an industrial environment, is unusual. Moreover,
it has also been successful, showing how simple tools and methodologies can
revolutionise industrial IT portfolio management techniques and practices.
References:


Chapter 8: Conclusion & Further Work

8.1 Introduction

This final chapter reviews the aim and objectives of the thesis, and the methods employed to undertake this research, before ending with a review of further work and modifications.

8.2 Restatement of Aim and Objectives

As stated in 1.6, the aim of the thesis was “to create and evaluate the effectiveness of a tool set for cost effective IS portfolio management.” Four objectives were identified that would complete this aim. These were:

1. To study successes and failures in IT/IS management, identifying the lessons learned, with particular reference to those pertaining to legacy systems.
2. To examine a number of established tools and methods for IS portfolio management at Rolls-Royce Naval Marine and determine if and how these tools can be adapted to better suit the Rolls-Royce environment. The success of these tools will then be evaluated.
3. To develop new tools and methods for use at Rolls-Royce Naval Marine and evaluate their success.
4. To test the general applicability of the tools and methods by applying the techniques to the wider Rolls-Royce company, or at least by gaining company acceptance to use these tools and methods in this wider context.

Some lessons learned (the first objective) had been identified during the first period of study (documented in chapter four). They were then used in both subsequent periods (documented in chapters five and six). The aim was served as the ‘scare stories’ alerted the management to the perils of uncontrolled IT portfolio management and also laid out some important lessons learned (see Appendix Two). Accomplishing the first objective does not fulfil the aim, but it does make fulfilment less uncertain, as lessons can be learned and mistakes not repeated. The legacy system problem at Rolls-Royce has benefited from these lessons learned, averting costly mistakes.
Many tools and methodologies were considered for use within Rolls-Royce (second objective, see Appendix One for documentation), but very few were actually chosen. Existing IT portfolio management techniques in the public domain were shown to be scant, and not fit for purpose. The need for something was identified, and the exact something was to be defined by business requirements in a grounded theoretic approach. It was seen that many of the tools were difficult to apply in a business context, requiring much in the way of data and computational time. There was a requirement for an array of simple tools and methodologies that could be used, rather than complex, mathematical models that were difficult to understand and change. Modification generally took the form of simplification, such as was the case with entity relationship modelling. The traditional form of this methodology can create highly complex models. But in this situation, the methodology was ‘slimmed down’ so that it was easy to use, describe and apply by people untrained in the discipline. The tools and methodologies are all developed in order to clarify management understanding of the state of the IT portfolio, and the possible impact and ramifications of any change. Being able to ‘see’ change impacts makes a very convincing argument.

The third objective leads on very naturally from the second, as there were some obvious gaps in the toolset to begin with. For example, a view of proposed architectures contrasted with existing architectures was not initially available. A simple diagram, either drawn on a piece of paper, or put together in MS PowerPoint, creates a powerful picture of a complex situation. Again, these tools and methodologies were used in Rolls-Royce and modified according to business requirements.

The fourth objective has been fulfilled – several components of the toolset are being evaluated for inclusion in the Rolls-Royce Quality Management System. Expenditure on legacy systems on the Naval Nuclear site of Rolls-Royce has decreased, but capability has actually been extended, with existing systems being used in innovative fashions, with various types of modification applied.
The four objectives have been fulfilled, and the thesis documents the creation and development of a toolset that has been successfully used in managing the IT portfolio of the Naval Nuclear business site of Rolls-Royce. The researcher feels that this creation of a toolset has shown a different approach to IT portfolio management than other works (as discussed in chapter two), being very much management and implementation focussed.

In the researcher’s opinion, the iterative approach was one of the key factors of the success of the entire project. It allowed the business representatives to take active ownership of the tools, and this then helped in the roll-out, as the business representatives felt that they had an interest in ensuring the success of the tools. The linkage to Checkland’s SSM is tenuous. However, the researcher feels that this link allowed the research framework to be employed, as the SSM approach had been used before, and appreciated.

A further key factor was the simplicity of the many tools and methodologies. The researcher has observed that the simpler tools and methodologies were the most successful and had the biggest impact. Developing simple tools is hard, as the situations they have to describe are often highly complex. However, it seems to be worth doing, as the tools and methodologies that have been suggested for inclusion in the RR QMS are all the simpler ones.

### 8.3 Further Work & Modifications

The researcher has accumulated many lessons learned during the course of study. The primary lessons learned was that simplicity is a singularly powerful argument. Without exception, the most successful tools and methodologies employed by the researcher were simple. Gaining management support of any project is crucial, and simple explanations aid this enormously. The temptation to use complex statistics and formulae should be resisted at all costs!

Secondly, financial modelling is a complex area, but many managers still use payback as a rough rule of thumb. The researcher acknowledges that financial modelling capability is not well represented in the current toolset. Various forms were used, and
payback was the best received. However, more financial modelling capability, significantly ones that present a simple answer, could be included. The existing financial modelling is simplistic – using payback as a measure of success. This is not a fault, as payback is often the ‘rule of thumb’ that many of the managers will use to judge the merits of a project. However, using more complex financial modelling (such as NPV, IRR, detailed cost-benefit analysis) would be a useful addition, as it would support the creation of a business case more effectively. Effective management of an IT portfolio constitutes effective financial management – the more accurate the modelling, the more effective the management and decision making can be, and the more options for change that can be considered.

Thirdly, the toolset has been developed by business requirements from the Naval Nuclear business site of Rolls-Royce. As such, it reflects this business, and the tools as they stand, may be too specialised for total transferral to a different business. However, the researcher believes that the concept of the tools and methodologies to be fully transferable, and the development process likewise. The researcher would like to use this toolset in other businesses, to see if similar problems and success are encountered.

A review of statistical modelling methods would also be desirable. The researcher did use regression analysis on defect data, but there was very little management interest in the study. The researcher would like to investigate the possibility of different types of statistical analysis, and examine whether more appropriate application of statistics would lead to better IT portfolio management.

8.4 Researcher Experience of Applying Software Engineering in Industry

As a general rule, the researcher found that management were predisposed to distrust software engineering techniques, and were very suspicious of any new technique, sometimes dismissing it as ‘the next fad’ or ‘bandwagon.’ This has been stated not just by Rolls-Royce, but also by several other managers at various corporations, all of whom seem to be disillusioned by software engineering’s ability to deliver tangible benefits. They recognise that many so-called bandwagons represent a drain on the company’s resources, and yet it would be catastrophic to miss the one that isn’t.
Much work needs to be done on improving the ‘image’ of software engineering. The author has been to many conferences where economics has been held up as an example of making molehills into mountains and vice versa. But it has also been the experience of the researcher that management hold software engineering in the same category as economics – ‘nice in theory, but useless in practice.’

There does seem to be several points of similarity between economics and software engineering. When the science of economics was initially being developed in the 19th Century, it was felt by many that economics could answer all the problems affecting countries, and that poverty would be eliminated by the careful manipulation of variables at both a macro- and micro-economic level. This proved to be impossible, forcing the school of economics to look very carefully at what it was and what it wanted to achieve. This period of introspection lead to the formation of many schools of thought and practice, but all recognised that there was an inherent disconnection between theory and practice [Gravelle & Rees, 1995]. Economics now looks at creating models that try to explain what might be happening at gross levels only. It has been realised that fine details are impossible to predict.

Information technology, or information science, seems to parallel the same process. There is still an expectation that sufficiently complicated models and tools might produce a universal panacea, ‘silver bullet’, ‘killer application.’ However, there is a growing disbelief that this is possible. Information technology is already beginning to fragment into many different disciplines, just as economics did a century ago. Many of the disciplines seem almost unrelated, for example formal methods and human-computer interaction. But both are valid sub-sets of information technology.

Perhaps what is need is a branch of Information Technology study that looks specifically at industrial application of software engineering and information needs. An approach is required that allows dissemination to industry, and solves real problems that face industry today.

This research has shown that there is much to be gained from a grounded theoretical approach based in industry. The grounded theoretic approach enforces practicality on
the development of the toolset – the tool must be fit for purpose, and the purpose is defined by the business. A tool that doesn’t provide concrete benefits and no improvements are suggested to improve the tool, is dropped with alacrity.

The industry-base meant that the researcher was submerged in the business, and the business requirements were an everyday phenomenon. They ‘loomed large’ in a way that is difficult to replicate in an academic environment. The researcher suggests that this aspect of ‘living’ the business requirements has long been recognised by consultants, who always work within the business when developing strategies and management plans, and rarely from an outside base.

The immediacy of these business requirements generated the imperative of simplicity. As has been mentioned before, simplicity has a key factor in the success of the research. A ‘pure’ academic approach may not have been able to visualise the benefits of simplicity, and may not have even considered some of the tools for inclusion, as they are very simple. However, the business impact of simplicity is hard to deny – the study had many successes, and some of the tools have been included in the Rolls-Royce Quality Management System.

A final point is an observation. The second phase of research acknowledges that the business representatives needed to develop trust not only in the tools and methodologies in the toolset, but also in the researcher. The business representatives needed to be certain that the researcher had adopted the business requirements as central to the development of the toolset, and did not use an ‘artificial’ approach, generated by academic interest in certain areas. The industry perception of academia seems to be poor – the term ‘academic question’ is used to describe a question that is interesting but not worth following up. An ‘academic answer’ is similarly one that is interesting but useless. With this distrust, it suggests that it would be very difficult to develop a theoretical approach to IT portfolio management, or even software management or engineering in general, and expect a full and fair trial. The researcher will have to prove not only the validity of their models/methods, but also their own validity in offering something that will improve a business process.
8.5 Overall Conclusion

It is the researcher’s opinion that the research was successful and worthwhile. Components of the toolset have been adopted internationally, across the whole of Rolls-Royce. The business unit that the studies were deployed was able to show a contribution to the ‘bottom line’ directly attributable to improved IT portfolio management, a contribution that was many times larger than the cost of improvements. The business unit was the only unit in the whole of Rolls-Royce that was able to demonstrate conclusively the requirements for IT investment, and consequently preserved the budget line, all in a time of industry-wide cutbacks in IT investment. The business unit has also been able to give a new lease of life to several systems that had previously been termed ‘legacy’ and viewed as a drain on the business.

From an experience point of view, this research has also been successful and worthwhile. It has shown that proactive IT portfolio management is possible, and that simple tools and methodologies can make a significant difference to a corporate approach to IT portfolio management.

It also suggests why there seems to be many software engineering techniques in academia, and so few actually employed by industry – the level of industry ‘distrust’ in ‘academic’ ideas not producing the required results. The immersion of the researcher within the industrial environment did much to alleviate the business representative concerns of an ‘academic approach.’
Bibliography


Anon, Course notes – Questionnaire Design, 1997

http://www.cc.gatech.edu/classes/cs6751_97_winter/Topics/quest-design/ Accessed 09/02


Bibliography


Boehm, B.W., Abts, C., Clark, B., Devani-Chulani, S. [1997] COCOMO II Model Definition Manual, University of Southern California


Boehm, B.W., Abts, C., Clark, B., Devani-Chulani, S. [1997] COCOMO II Model Definition Manual, University of Southern California


BS IEC 61649:1997


Campbell, I.[1996], The Intranet: Slashing the Cost of Business, International Data
Bibliography

Corporation, Framingham, MA.
Computer Weekly [2001], “Staffing issues top directors’ agenda”, 26 July 2001, pg. 18
Cushing, K. [2002] “Stem the flood of IT waste,” Computer Weekly, 18/07/02, pg.22
Bibliography


221
Bibliography


Grage, P, Introduction to JDBC

http://shrike.depaul.edu/~pgrage/dist_sys/ds520_p1.htm Accessed 08/00


Longmans, Harlow, UK


Hegel, G.W.F [1821] Grundlilnen der Philosophie des Rechts oder Naturrecht und Staatswissenschaft im Grundrisse (Foundations of the Philosophy of Right: Natural Right and the Political Science in Outline), University of Heidelberg. Translation by Inwood, M. Penguin, London


Järvinen, P. [2001] On Research Methods, Juvenes-Print, Tampere


Accessed 08/00


King, J. [2002] “Make IT bigger, better, faster,” Computerworld, 01/01/02


Bibliography


Koch, M [2001]. “Leverage legacy systems with a blend of XML, XSL, and Java,” ITworld.com, 26/01/01


Likert, R. [1932], “A Technique for the Measurement of Attitudes”, Archives of Psychology, 140(6), 1932
Miller, G.A. [1956], “The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information,” Psychological Review, Vol. 63, 1956,
Bibliography

pp. 81 – 97


Oracle Corporation Oracle 8i Concepts
http://www.technet.oracle.com/doc/server.805/a58227/ch_repli.htm Accessed 01/01


Pettigrew, A.M. [1985] “Contextualist research: A natural way to link theory and practice” in Lawler, E., Mohrman, A., Mohrman, S., Ledford, G., Cummings, T (eds.) Doing research that is useful for theory and practice, Jossey-Bass, San-Francisco
Bibliography

Phoenix Group, *Legacy Systems Wrapping with Objects*, URL:
http://www.phxgrp.com/jowdewp.htm Accessed 01/01
Bibliography


Bibliography


Udell, J. [2002] “Java, XML, and Web Services”, *Infoworld*, 25/03/02
WebMethods, B2B Integration: The Drive to Gain and Maintain Competitive Advantage, July 1999,
Yin, R.K. [1989] Case study research: Design and Methods, Sage Publications, Beverly Hills, California

230
Appendix I: Estimation and Evaluation Measures and Models

The three most dangerous things in the world are a programmer with a soldering iron, a hardware type with a patch, and a user with an idea."

Rick Cook

"What is strategy? There is only one universally accurate answer – it means exactly what you want it to mean to make your point"

Jo Owen

Appendix I.1 Introduction

Software cost and effort estimation models were considered as part of the research – the ability accurately estimate some of the costs involved in creating new software, or modifying existing software would be a useful addition to the toolset. The approach used to consider the cost models was two fold. Firstly, freely available models were considered for applicability to the situation faced by Rolls-Royce. The assessment was conducted by the researcher and selected business representatives. The second stage required the assessment of the commercially available models. The commercially available models were considered to be superior to the freely available models, but attracted an investment requirement. The second phase would only be entered if the results of the first stage were favourable.

Below are some of the freely available cost models considered. The selection was dictated primarily by the fact that all of the cost models have commercially available counterparts that are more complex and favourable.

Appendix I.2 Effort And Cost Estimation Models

The next section introduces a range of cost and effort estimation models that were considered.
I.2.1 1965 SDC Cost Model

As described by Boehm [1981], the Software Development Company model is a fifteen parameter, linear regression estimation, taking the form:

\[ MM = \beta_0 + \beta_1(\text{Lack of Requirements}) + \beta_2(\text{Stability of Design}) + \beta_3(\text{Percent Math Instructions}) + \beta_4(\text{Percent Storage/Retrieval Instructions}) + \beta_5(\text{Number of Subprograms}) + \beta_6(\text{Programming language}) + \beta_7(\text{Business application}) + \beta_8(\text{Stand-alone Application}) + \beta_9(\text{First Program on Computer}) + \beta_{10}(\text{Concurrent Hardware Development}) + \beta_{11}(\text{Random Access Device Used}) + \beta_{12}(\text{Difference Host, Target Hardware}) + \beta_{13}(\text{Number of Personnel Trips}) + \beta_{14}(\text{Developed by Military Organization}) \]

Equation A.1: Original SDC Cost Model

The model was developed for military software projects and so was considered for use, given the context of the software development project. However, it is noted that many of the variables reflect the era of computing that the model was defined in (the 1970s). Particularly the number of math instructions, personnel trips and the access devices used variables would seem antiquated, and not directly applicable to modern software design. It will be considered for further evaluation principally to the emphasis placed on design for military organisations, and has been adapted further to suit changing requirements.

Appendix I.3 Putnam SLIM model for Size

Putnam’s SLIM (Software Lifecycle Model) was first developed in the 1970s [Putnam, 1978]. The model has since been extended and updated, as it is owned by Quantitative Software Management, Inc. It is a constraint model for projects with an estimated size of greater than 70,000 lines of code (70 KLOC). The model assumes that effort is distributed in a similar fashion to a collection of Rayleigh curves. Equation A.2 shows the freely available SLIM model, employing a basic effort macro-estimation:

\[ S_S = C_4K^{1/3}t_d^{5/3} \]

Equation A.2

where:

- \( S_S = \text{number of delivered source instructions} \)
The Rayleigh curve represents manpower as a function of time. Norden noted that the Rayleigh distribution provides a good approximation of the manpower curve for several hardware development processes [Pillai and Sukumaran Nair, 1997].

Putnam's SLIM can estimate the technology constant as a function of the project's use of modern programming practices, hardware constraints, interactive development, and other factors.

The SLIM model includes useful extensions such as:

- Manpower distribution
- Cash-flow
- Major-milestone schedules
- Reliability levels
- Computer time
- Documentation costs

On balance, the SLIM model approach has provided a number of useful insights into software cost estimation, including:

- The Rayleigh-curve distribution for one-shot software efforts
- The explicit treatment of estimation risk and uncertainty
- The cube-root relationship defining the minimum development time achievable for a project requiring a given amount of effort

For these reasons, use of Putnam's SLIM approach can be considered for specific application. Ndaiye [2000] found SLIM to be very accurate, but only for tightly specified projects that had certain characteristics, such as size greater than 70 KLOC, and not modularised. She also noted that the model did not cater particularly well for modern object-oriented languages. However, this was an older version of the software produced by Quantitative Software Management Inc.
1.3.1 Doty model for Effort estimation

The Doty model utilises a regression-based approach, with thousands of lines of code (KLOC) as a primary input [Boehm, 1981]. Equations A.3 and A.4 show the models for general application:

\[ MM = 5.288(KDSI)^{0.047} \]  
for KLOC \( \geq 10 \)  
\[ \text{Equation A.3} \]

\[ MM = 2.060(KDSI)^{0.047} \left( \prod_{j=1}^{14} f_j \right) \]  
for KLOC < 10  
\[ \text{Equation A.4} \]

where KDSI = 1000 delivered source instructions (delivered source instructions are similar to non-comment lines of code), and MM is Man Months.

It possesses the ability to reflect different factors in the design. The high level simplicity of the equation is immediately appealing, especially give that the alternative to cost models is expert judgement, which often gives a fairly accurate answer in a short space of time. The Doty model is used in some commercial programs.

1.3.2 COConstructive COSt MOdel (COCOMO) and COCOMO II

COCOMO is a constraint model, demonstrating the relationship over time, between a number of parameters of effort. It was first published in 1981, after analysis of a number of projects at TRW [Boehm, 1981].

COCOMO is a hierarchy of three increasingly detailed models which range from a single macro-estimation scaling model as a function of product size (the originally published COCOMO model [Boehm, 1981]), to micro-estimation model with a three-level work breakdown structure and a set of phase-sensitive multipliers for each cost driver attribute.

All of the models take the same basic form:

\[ E = aS^b \times EAF \]  
\[ \text{Equation A.5} \]
where E is the effort measured in person-months, S is the size measured in KLOC, and EAF is the effort adjustment factor. Factors a and b depend upon the development mode. Boehm [1981] defined three:

1. Organic – relatively simple projects, with small teams working to a set of informal requirements
2. Embedded – a project that must operate within a tight set of constraints
3. Semi-detached – intermediate project with elements of both organic and embedded modes. Mixed teams work to a set of rigid and another set of more informal requirements

The Basic COCOMO model is a simple function of estimated program size and adjustment according to the development mode. The EAF from Equation A.5 is equal to one. The intermediate model encapsulates the basic plus a set of effort multipliers determined from the product’s ratings on a set of 15 cost driver attributes. The estimated development is obtained by multiplying the nominal effort estimate by all of the product’s effort multipliers. Additional factors can be used to determine dollar costs, development schedules, phase and activity distributions, computer costs, annual maintenance costs, and other elements from the development effort estimate.

COCOMO II is a modification of the COCOMO process, reflecting the changes in software development practice. COCOMO was developed using data from bespoke programming projects. By the time COCOMO II was released, software development had become more commoditised, with commercial off the shelf applications and application development utilities being very common. COCOMO was updated to reflect this change [Boehm et al, 1997]. COCOMO II includes the Application Composition Model for early prototyping, the Early Design Model, and the Post-Architecture Model, each embodying progressively more detail.

A significant change in the application composition model is the use of object point, rather than lines of code. Object points are derived through analysis of the number of screens, reports and third-generation components used, or will be used, in the application. However, KLOC are used in the other two models.
### 1.3.3 Other Models Considered

Three other models have been considered, but not carried forward for further application. These are the TRW Wolverton model, the Bailey-Basili meta-model, and an application of Halstead’s software science. They were briefly considered, but dropped by the researcher for a variety of reasons developed below.

The TRW Wolverton model [Boehm, 1981], developed in 1974 as a precursor of the COCOMO model mentioned above, shows a number of curves of software cost per object instruction as a function of relative degree of difficulty (0-100), novelty of application (new or old), and type of project. The model is well calibrated to a class of near-real-time government command and control projects, but is less accurate for some other classes of project. Boehm [1981] considers the model to provide a good breakdown of project effort by phase and activity. The context in which the model is used, namely government/military projects, made the selection of this model appealing, but some of the variables of the original model are outdated and would have to be updated, and the majority of the data would be not available in the specific application of the model. The intention is to use models with as little modification as possible. The updating of the TRW model would be of significant, with many variables being dropped and several new variables added in their place. Boehm, [1981] notes that lessons learned from this model were applied in the development of the COCOMO models. Given that dated nature of the model, the inclusion of the more advanced COCOMO model, and the lack of an appropriate commercially available model, this model was dropped from consideration.

The second model considered is the Bailey-Basili Meta-Model [Bailey and Basili, 1981]. It is derived from the scaling equation:

\[
MM_{NOM} = 3.5 + 0.73(KDSI)^{1.16}
\]

Equation A.6

where MM represents man months of effort, and KDSI is thousand delivered source instructions.

In this model, Bailey and Basili used two additional cost-driver attributes (methodological level and complexity) to model the development effort of projects in
the NASA-Goddard Software Engineering Laboratory. The simplicity was immediately appealing. However, this simplicity is misleading, and the constants are not considered stable for different sizes of programs [Boehm, 1981], and so the application of this model would have been problematic, given a problem concerning multiple programs to evaluate.

The third model is a derivation of Halstead's complexity metric. A computational formula is given by Curtis [1980]:

\[ E = \eta_1 N_1(N_1 + N_2) \log_2(\eta_1 + \eta_2) \]

\[ 2\eta_2 \]

Equation A.7

where

\[ \eta_1 = \text{number of unique operators} \]
\[ \eta_2 = \text{number of unique operands} \]
\[ N_1 = \text{frequency of operators} \]
\[ N_2 = \text{frequency of operands} \]

The units of \( E \) are the number of elementary mental discriminations needed to understand \( P \). Stroud has asserted that the human mind is capable of \( \beta \) mental discriminations, where \( 5 \leq \beta \leq 20 \). Miller [1956] stated that the human mind is capable of rather less than the Stroud number, where \( \beta \) is equal to seven, plus or minus two. Halstead originally used \( \beta = 18 \), and thus the time, \( T \), required to code program \( P \), of effort \( E \) is:

\[ T = \frac{E}{18 \text{ seconds}} \]

Equation A.8

There are several significant problems with Halstead's metric, beyond the methodological problems identified above. The first is that the model depends upon completed code. Use of estimations significantly increases the inaccuracy of the model. The second is that the predicative capabilities of the model are limited [Anon, 2002]. The last problem is the use of Stroud's number. Stroud's research was predicated upon the use of colour to distinguish between mental discriminations. Miller's research used discrete data. But the application of both numbers to complex cognitive activity, such as programming, has been disputed, initially by Curtis [1980].

1.3.4 Regression Models
There has been much work conducted on using regression techniques, and applying
them to the problem of software maintenance effort estimation. An overview of
software fault analysis is given by Musa et al [1990], but more specific models are
presented by Jelinski and Moranda [1972], and Schick and Wolverton [1978]. Graves
et al [2000], conducted a series of tests on fault prediction models, ranging from the
stable fault incidence model, through various models using generalised least squares
(a less restrictive version of ordinary least squares described above), to a weighted
time damp model. They found that the weighted time damp model was superior to all
the other models, but there was still a substantial degree of uncertainty to predictions.

Appendix I.4 Financial Evaluation Measures

1.4.1 Internal Rate of Return

Like NPV, the internal rate of return (IRR) is a discounted cashflow analysis metric
utilising the concept of time value of money. The IRR is also referred to as the
discounted rate of return [Drury, 2000]. The IRR is the interest rate, $K$, that when
used to discount all cashflows resulting from an investment, will equate the present
value of incoming cashflow to that present value of the outgoing cashflow. More
simply, it is the rate, $K$, that will give an NPV equal to zero. Solving Equation 7.34
for $I_o$ (as shown in equation 7.36), will give the IRR for a project, rather than the
NPV:

\[ I_o = \sum_{n=1}^{n} \frac{FV_n}{(1 + K)^n} \]  \hspace{1cm} \text{Equation A.9}

Drury [2000] states the NPV and IRR metrics should lead to the same decisions, as
long as the decision rules stay constant, and the cashflows are conventional (initial
cash outlay, followed by a series of cash inflows). However, if the projects are
mutually exclusive, the IRR and NPV metrics can disagree. Drury presents a situation
where the IRR can give an incorrect solution, due to its methodological construction
[Drury, 2000]. Furthermore, the IRR metric can also not easily differentiate
percentage returns as effectively as the NPV metric.
In addition to these problems, the IRR metric has significant difficulties in adjusting for unconventional cashflows. The construction of the IRR metric requires that each negative cashflow will have an associated IRR. If there are multiple negative cashflows for one project, that project would by definition have to have an IRR for each negative cashflow, the multiple IRR metrics would lead to a nonsense figure.

1.4.2 Return on Investment

Return on investment (ROI) (also known as the accounting rate of return and return on capital employed [Drury, 2000]) is one of the most common investment appraisal techniques, as mentioned by King [2002] and Vowler [2002]. The ROI is given by [Drury, 2000]:

\[
\text{ROI} = \frac{\text{average annual profits}}{\text{average investment}}
\]

Equation A.10

ROI has a serious disadvantage in that it takes no account of the time value of money. This results in a bias towards projects with front-loaded returns. Further problems with using ROI are highlighted by Ruben [2002], including the problem with user bias, double counting of current profits with proposed future profits, inherent problems with over-enthusiastic predictions of future profits, and the fact that ROI does not necessarily indicate whether the investment is in support of business strategy, and is integrated with all other plans, such as financial, operational, IT, and human resources.
1.4.3 Cost, Benefit, and Effort Measures – Summary

Objective three stated that several tools and measures had to be used in order to develop an IT strategy. Cost, benefit, effort and quality are suggested as cardinal measures for industry. This chapter has explored many different ways to measure cost, benefit, and effort. Quality standards and measures were covered in Chapter Five.

Measuring effort requires that size be estimated. Lines of code are, de facto, an acceptable measure of size, although it has many acknowledged shortcomings. Function points are possibly superior to KLOC as a measure of size, certainly from a theoretical viewpoint, but require more data and time to measure than KLOC. Even once a measure of size is agreed, it is hard to prove that any effort estimation model using either KLOC or function points is superior to expert judgement.

Cost measures suffer from a similar problem. The first issue is that a large factor of cost is effort, and errors in effort estimation automatically follow through and are sometimes magnified. The second problem is that there are so many models, and very few of the models seem to measure the same set of costs. Sweet [1996] acknowledges that software cost estimation is not perfect, but qualifies that statement by stating that it is a very useful exercise, and can significantly reduce costs and increase software quality.

Many of the costs covered by King et al [1978] are covered by some of the models, but many are not, and are considered out of scope, such as rental costs, depreciation costs, and some forms of overhead costs. Through life costing, however, does include these costs. But it does suffer from a lack of accuracy, but the theory requires a different set of assumptions to the other cost models, and is intuitively more acceptable. It also produces a single cost figure, with error margins, and is not amenable to detailed disaggregation, which would show the exact contribution of each of King’s identified costs. It should be noticed that some disaggregation is possible.
Appendix I.5  Summary

This appendix has presented some of the cost and effort estimation models available, and looked at some of the financial evaluation measures that are commonly applied to projects.
Appendix I

References:


Boehm, B.W., Abts, C., Clark, B., Devani-Chulani, S. [1997], COCOMO II Model Definition Manual, University of Southern California


King, J. [2002] “Make IT bigger, better, faster,” Computerworld, 01/01/02


Pillai, K., Sukumaran Nair, V.S. [1997], “A model for software development effort and cost estimation,” IEEE Transactions on Software Engineering, 23(8), pp. 485-497


243
Ruben, S. [2002] “Don’t use ROI in investment decisions”, Computer Weekly, 07/03/02, pg. 26
Accessed 06/02
Appendix II: Quality In-Depth

Appendix II.1 Introduction

This appendix is divided into two sections. The first section introduces the concept of 'quality' and methods that have been put forward to achieve quality outputs. The second section recounts various measures of quality.

Appendix II.2 Approaching Quality from a Holistic Viewpoint

Quality is a continuing concern of businesses the world over. The industrial revolution in the late 19th Century showed the advantages of continually improving productivity and quality. Capitalism provided the Darwinian impetus — those companies that managed to improve became successful. Modern manufacturing process improvement began after World War Two with W. Edwards Deming's work with the Japanese during the 1950s. He pioneered manufacturing quality control methods in Japan, and created what later became known as Deming's Fourteen Points [Walton, 1986]. His methodology had an enormous impact on the Japanese, and their rapid adoption of process improvement contributed to their rise as one of the world's foremost manufacturing countries during the 1950s, 60s, 70s and 80s.

Deming introduced the same methodologies to the United States during the 1980s with an NBC documentary. Companies were eager to adopt this methodology that had made significant inroads into their market shares. Early adoptees included the Ford Motor Company and AT&T [Simmons et al, 1997].

II.2.1 Software Engineering Quality - The SEI and CMM

Manufacturing process improvement methodologies have formed the kernel of software process improvement. During the late 1980s the US Department of Defense initiated software process improvement programs, based around the works of Deming, Crosby and others [Bollinger and MacGowan, 1991]. The result was the Software Engineering Institute (SEI) Software Process Assessment Program. The principle of the assessment program was to assess the various interacting elements of software
Appendix II

production, such as tasks, people, tools, standards, and other resources. The SEI also produced a Software Capability Evaluation program in parallel.

Both the Software Process Assessment and the Software Capability Evaluation are based upon the Software Process Maturity Framework, described by Watts Humphrey in 1989 [Humphrey, 1989]. The principle is to introduce continuing improvement in the production of software products. The process maturity framework has 5 levels, from a basic level, where software development projects are not optimised and success is rarely repeatable, to level 5, where software project successes are predictable and optimised.

Humphrey, drawing from Deming’s 14 points, espoused six fundamental principles:
1. Major changes to the software process must start at the top
2. Ultimately, everyone has to be involved
3. Effective change requires a goal and knowledge of the current process
4. Change is continuous
5. Software process changes will not be retained without conscious effort and periodic reinforcement
6. Software process improvement requires investment

All of these have lead to the Capability Maturity Model (CMM), being applied both to software projects, and more recently to generic development projects. There is currently great debate about the efficacy of CMM. However, it is not the only development model for achieving software quality.

II.2.2 The PRINCE Methodology

PRINCE (Projects IN Controlled Environments), and the subsequent update PRINCE 2, are project management methodologies aimed specifically at project organisation, management, and control. PRINCE was originally developed for IT projects, but has been used on many non-IT projects. The initial development was by the UK’s Central Computer and Telecommunications Agency (now part of the Office of Government Commerce). It is now the de facto standard for project management in the UK public sector [Goodwin, 2001].
PRINCE2 is a process-based approach to project management, allowing tailored, flexible, and scaleable solutions for project management. The individual processes are defined with key inputs and outputs and specific objectives and activities that need to be performed.

PRINCE 2 enables projects to have:
1. A controlled and organised start, middle and end
2. Regular reviews of progress against plan and against the Business Case
3. Flexible decision points
4. Automatic management control of any deviations from the plan
5. Involvement of management and stakeholders at the right time and place during the project
6. Good communication channels between the project, project management, and the rest of the organisation

II.2.3 Dynamic Systems Design Method (DSDM) and Joint Application Development (JAD)

These methodologies have a more specific focus than SSADM. It is focussed much more on rapid development, where projects have short timescales. JAD was originally developed by IBM in the 1970s to increase user involvement and buy-in of software development. It was then revived in the 1990s as part of the drive towards rapid application development and time-boxing. DSDM was developed by a consortium of supplier and customer organisations that required a standard approach to the development of rapid application development projects.

II.2.4 Unified Modelling Language (UML)

UML is a relatively new specification standard. It has swiftly become a de facto standard for object-oriented analysis and design. It combines a set of specification and design notations for object-oriented systems. It was first launched in 1996, and endorsed by the Object Management Group in 1997. Subsequently, it has attracted support from significant IT companies, including Hewlett-Packard, Oracle, and Microsoft [Goodwin, 2001].
Its significant strength lies in its comprehensive ability to communicate and represent the development project to developers and stakeholders through a selection of graphical tools. However, it does not provide and project management tools in itself. However, packages that utilise UML quite often have some project management capability. Making sure that users and developers are looking at the project in the same way and in the same language is a very big advantage. Communication has always been a critical success factor of any project.

However, the major perceived weakness with UML springs directly from its strength. The graphical views, such as entity relationship diagrams, case diagrams and data flow models, are very valuable and give great insight. But UML has nine graphical views, and there are some questions over whether this plethora of views creates more confusion that enlightenment.

### II.2.5 Six Sigma

Six sigma is a quality philosophy, based principally at reducing costs at the same time as increasing service. It is a rigorous, quantitative approach to quality improvement, utilising a number of statistical tools and creative techniques to form a holistic quality improvement approach.

### II.2.6 Total Quality Management

Total quality management (well-known by its acronym, TQM), embodies an holistic approach to quality. It looks at the organisation as a whole, analysing each activity, and improving it [Oakland, 1995].

The primary impact is to introduce awareness of quality at all levels and in all things. The management develops a strategic overview of 'quality,' working on problem prevention. The total quality paradigm is to exploit quality at each stage of the process. With high quality at every stage, the quality of the eventual output must, perforce, be of high quality.
TQM recognises that quality quite frequently involves a complete change in the working culture of a company. Many companies employ quality assurance of the final product, without looking at the process. The implementation of TQM has to radically change the way of quality assurance, from examining the final product, to examining all stages [Oakland, 1995] [Turner, 1993].

Appendix II.3 Measures of Quality

II.3.1 Usability

Customer satisfaction and the through-life cost are much affected by usability. Fenton and Pfleeger [1997] define usability as “The probability that the operator of a system will not experience a user interface problem during a given period of operation under a given operational profile.”

Boehm et al [1978], informally define usability as “[the] usability of a software product is the extent to which the product is convenient and practical to use.” This definition, in contrast to Fenton & Pfleeger’s, is much more intuitive, and is probably better known as “user-friendliness.”

However, neither of these two definitions actually help in formulating a specific measure for usability. For example, how would one define a user interface problem, in order to derive the probability distribution for use with Fenton and Pfleeger’s definition? Similarly, Boehm’s definition would include aspects of:

- Well-structured menus
- Intuitive graphics
- Well-structured manuals
- Informative error messages
- Consistency throughout the user interface
- Useful help functions

However, this raises the question of how to measure these aspects. This creates a quandary – the attributes of usability are extremely difficult to measure, and could only be used to measure usability with great difficulty. Given this, decomposition of
usability to more fundamental principles is required. COQUAMO [Boehm, et al. 1978] provides automated support for measuring usability and setting targets, through the use of user templates. There exists the problem of user-bias, however, the targets and measurements face exactly the same bias, and thus the targets can be considered valid.

11.3.2 Weibull Analysis and Derivation

The technique is based around the Weibull distribution, which is a two-parameter distribution described by a shape parameter, $\alpha$, and a scale parameter, $\beta$, using the following form of probability density function (equation A.1) and a cumulative distribution function (equation A.2):

$$f(t, \alpha, \beta) = \frac{\alpha}{\beta^\alpha} t^{\alpha-1} e^{-\left(\frac{t}{\beta}\right)^\alpha}$$  \hspace{1cm} \text{Equation A.1}

$$F(t, \alpha, \beta) = 1 - e^{-\left(\frac{t}{\beta}\right)^\alpha}$$  \hspace{1cm} \text{Equation A.2}

The effect of different values of $\alpha$ can be shown in figure A.1:

![Figure A.1 Cumulative Probability Distribution Graph for Two Values of $\alpha$](image)
If it is assumed that failure data can be satisfactorily described using a Weibull distribution, then estimates of the parameters $\alpha$ are derived from equation A.3, and $\beta$ from equation A.4:

$$
\alpha = \left[ \frac{\sum_{i=1}^{r} t_i^{\alpha} \ln(t_i) + (n-r)T^{\alpha} \ln(T)}{\sum_{i=1}^{r} t_i^{\alpha} + (n-r)T^{\alpha}} \right] - \frac{1}{\alpha} \sum_{i=1}^{r} \ln(t_i) = 0 
$$

**Equation A.3**

$$
\beta = \left( \frac{1}{r} \left[ \sum_{i=1}^{r} t_i^{\alpha} + (n-r)T^{\alpha} \right] \right)^{\frac{1}{\alpha}}
$$

**Equation A.4**

In both equation A.3 and equation A.4, $n$ is the sample size, $r$ is the number of failures within the sample, $(n-r)$ is the number of censored items, $t_i$ is the time to the $i^{th}$ failure, and $T$ is the analysis time.

It is also possible to calculate confidence boundaries of the fit parameters obtained from equations A.3 and A.4. For the interval estimation of parameter $\alpha$, the multiplying factors $w_1$ and $w_2$ should be calculated, as in equation A.5:

$$
w_1 = \left[ \frac{k_1}{rC} \right]^{\frac{1}{1+q^2}} \quad \text{and} \quad w_2 = \left[ \frac{k_2}{rC} \right]^{\frac{1}{1+q^2}}
$$

**Equation A.5**

Where:

$$
k_1 = \chi^2_{\frac{r}{2}}[(r-1)C] \quad \text{and} \quad k_2 = \chi^2_{\frac{r}{2}}[(r-1)C]
$$

**Equation A.6**

When,

$$
C = 2.14628 - 1.361119q
$$

**Equation A.7**

Where: $q = \frac{r}{n}$ and $\chi^2_{p}(v)$ is the $p$ fractile of the $\chi^2$ distribution with $v$ degrees of freedom.
The confidence limits of the $\beta$ parameter are calculated using equations A.8 and A.9:

\[
d_1 = \frac{A_3 + x \sqrt{x^2 (A_2^2 - A_4 A_5) + r A_4}}{r - A_3 x^2} \quad \text{Equation A.8}
\]

\[
d_2 = \frac{A_3 + x \sqrt{x^2 (A_2^2 - A_4 A_5) + r A_4}}{r - A_3 x^2} \quad \text{Equation A.9}
\]

With

\[
A_6 = 0.029 - 1.083 \ln(1.325 q)
\]

\[
A_5 = 0.2445 (1.78 q)(2.25 + q)
\]

\[
A_4 = 0.029 - 1.083 \ln(1.325 q)
\]

\[
A_3 = -A_6 x^2
\]

where $x = U_{\left(\frac{1-p}{2}\right)}$ and $U_p$ is the $p$ percentile of the normal distribution.

This gives the confidence interval for $\beta$ as equation A.10:

\[
[ A_1 \beta : A_2 \beta ] \quad \text{Equation A.10}
\]

With (equation 5.12 and 5.13)

\[
A_1 = \exp\left(\frac{-d_1}{\alpha}\right) \quad \text{Equation A.11}
\]

\[
A_2 = \exp\left(\frac{-d_2}{\alpha}\right) \quad \text{Equation A.12}
\]

The mean time to failure, $\overline{m}$, can be obtained using equation A.13,

\[
\overline{m} = \beta \Gamma\left(1 + \frac{1}{\alpha}\right) \quad \text{Equation A.13}
\]

where $\Gamma(z)$ is the gamma function of $z$.

### II.3.3 Through-Life Cost

Through-life cost applicable as a quality indicator, as the cost of an application is very much dependent upon its quality. Because through-life costing reduces all aspects of the project to a single cash figure, it can be used as a quality metric [Bradley and Dawson, 1998].
As mentioned earlier, through-life costing has the advantage that it models costs over the entire lifetime. Given that the majority of software costs occur after development and implementation [Bradley and Dawson, 1995] [Bradley and Dawson, 1996], through-life costing can be considered an apt indicator of the ‘total’ quality of an application.

Appendix II.4 Holistic Quality Models

Holistic quality models recognise that quality is difficult, if not impossible, to measure with a single figure. The models use a ‘basket’ of indicators to demonstrate what constitutes a ‘quality’ outcome, and what does not.

II.4.1 McCall’s Model and ISO 9126

McCall’s model is a fixed quality model, focused on executable code, and identifies 11 key quality attributes from the user perspective [McCall, 1977]. These quality attributes, or quality factors, are hierarchical. The high-level external attributes, such as usability, testability, and interoperability (amongst others) are then decomposed into various criteria which can then be measured. For example, the usability factor is decomposed into operability, training, communicativeness, I/O volume and I/O rate criteria. Further decomposition is performed on the criteria to produce quality metrics. The McCall model identifies 25 criteria and 41 metrics.

McCall’s model formed the basis of an international model for software quality – ISO 9126. Under this standard, quality is interpreted as being “The totality of features and characteristics of a software product that bear on its ability to satisfy stated or implied needs.” Thus defined, the quality aspect is then apportioned into six factors:

1. Functionality
2. Reliability
3. Efficiency
4. Usability
5. Maintainability
6. Portability
Similar to McCall's model, these factors are further decomposed. The standard aims to be comprehensive. There have been some criticisms of the standard, but several companies have used the standard successfully.

II.4.2 Boehm Model

Boehm's model [Boehm et al, 1997] was produced about the same time as McCall's. Boehm et al produced a fixed quality model, just as McCall did. However the Boehm model looked at quality constructs, and decomposed these down further, producing intermediate and primitive constructs. Various software metrics were then used to measure these primitive constructs. Boehm et al identified seven intermediate constructs, such as reliability and efficiency. These were then further decomposed into twelve primitive constructs.

II.4.3 Define-Your-Own Models

This approach uses consensus amongst users to create specific models for a specific occasion. The consensus decides which quality attributes are important to them, and then decides the appropriate decomposition. This method has been used by Kitchenham and Walker [1989], building on work done by Gilb. Gilb's method was to design software with 'measurable objectives,' using a philosophy of evolutionary development. The development team produces the product incrementally, based on priorities set by the user on the types of functionality being delivered. The priorities are assigned by the user identifying key software attributes, described in quantifiable terms. This sets measurable objectives for the development of the software. Kitchenham and Walker used this method as the basis of COQUAMO (Constructive Quality Model). This took Gilb's ideas and produced an automated tool as part of an ESPRIT project.

II.4.4 EFQM

The EFQM (European Framework for Quality Management) is a Brussels-based organisation, dedicated to achieving quality in its members and partner organisations [EFQM, 2002]. The EFQM was founded through the actions of 14 major European companies (Bosch, BT, Bull, Ciba-Geigy, Dassault, Electrolux, Fiat, KLM, Nestlé, Olivetti, Phillips, Renault, Sulzer, and Volkswagen). The European Commission was
also involved. From these beginnings, the EFQM has grown to have more than 800 members.

The aim was to develop a European alternative framework, to parallel the developments in America (Baldrige Model [Simmons et al, 1997]), and Japan (Deming). The European Model for Business Excellence Model (now better known as the EFQM excellence Model) was introduced in 1991 as the framework for organisational self-assessment.

EFQM makes extensive use of the principles of TQM in order to foster a desire and reputation for quality within European businesses [EFQM, 2002]. It is one way in which TQM can be applied to a company through the use of a structured template. This is a very popular approach, especially in governmental organisations.

Appendix II.5 Single Aspect Measures of Quality

Despite the criticism that quality is difficult to assess using a single measure, there are many companies that employ a single measure of quality. Below are some of the ones that are used. It is recognised that some companies use a collection of these metrics, but in ways that are not in line with the holistic models detailed above.

Appendix II.6 Summary – Quality Software and Software Quality

It has been shown that there are many standards, methods and measures for quality. Given the existence of so many quality-ensuring methods, why are so many quality failures reported? Is it possible that there is a difference between software quality and quality software?

It seems that the term ‘software quality’ is most commonly used when referencing an internal aspect of the code – reliability, efficiency, reusability, and maintainability amongst others. As has been shown above, there are many frameworks, standards and models that can be used to develop software with a high internal quality.

However, it would seem that ‘quality software’ refers to more ‘external’ aspects of the software – how well does the software serve business needs? Can the software fulfil
what is required (given that what is defined may not be what is required)? There is considerable confusion in software engineering over the exact definition of technical concepts and terminology. As Xia [2002] noted, there seems to be a distinct lack of pure software engineering theory, with a disconnection between theory and application. In Xia's opinion, there is currently "a lack of nominal and real definitions in SE (software engineering)... which contributes directly to the immaturity of the discipline."

Setting aside the theoretical objections to these models and assuming that the models do have some merit, it does raise an interesting question. Given the high level of reported IT system implementation failures, are these models and standards being used? There are three possible answers to that question. The first is that the standards, methods and measure mentioned above are not being used, and this would in part explain the number of reported failures. The second is that the theoretical objections to the models are valid — since the terminology has not been rigorously defined, the models are flawed and thus are not performing their function correctly. This is a possible answer, and one that would be hard to prove either way. The third answer is that the standards are being applied incorrectly.

Given that CMM has thousands of companies certified and registered at CMM Level 3, and the many companies registered as being ISO 9000 certified, the first answer is indefensible. There are too many companies with quality standard certifications for quality standards to not be employed and enforced. This is not to say that every company is using quality standards and methods, but that awareness of quality standards and methods is high. The second answer is intuitively correct, but it does not seem to explain everything; the same argument can be applied to the last answer. There could be other explanations.

One of these further factors could be the existence of an IT 'arms race.' Companies may invest in software simply because their competitors are, and they cannot risk their competitors gaining too much of an advantage. The rapid uptake of technologies such as enterprise resource planning, customer relationship management, and virtual private networks, is indicative of cut-throat competition. Once one company demonstrates a successful implementation (i.e. achieves dramatic returns on
investment), other companies are bound to follow, as they cannot risk other companies also having a success, whilst they themselves have nothing.

Another explanation could be that there are incorrect assumptions about the methods and standards. For example, there are many reported companies at CMM level 3, but much less at levels 4 and 5. It would seem that there is an industry opinion that proceeding to higher levels is not cost effective. There is also the perception that CMM is for 'large companies,' a perception which is reinforced by the costs of certification.

PRINCE and PRINCE 2 are widespread in the UK, but largely confined to the Government and Government suppliers. But it is not a guarantee of success. The British government has mandated use of PRINCE and PRINCE 2 on many projects, and the success rate of UK Government IT projects is notoriously poor. The focus of these methods is on development and implementation. There is little in the way of maintenance, unless it is managed as a separate project.

SSADM is detailed, but it is possible that it could be overly so. Many projects exist in 'Internet time' where the priority is to be first to market. The rigour imposed by SSADM could be perceived to be harming the success of the project by extending timescales. Again, the same criticism of PRINCE can be levelled at SSADM - it does not explicitly take the full cost and requirements of the software project into account.

DSDM and JAD again focus on project delivery, but in rapid time, as opposed to SSADM. But there does not seem to be widespread use of these methods, and again it does not address the full impact of software on the business.

Six sigma again has a wide uptake, but may be seen as too constrictive for smaller companies that are focused on being first to market.

UML, similar to the other methods has many positive aspects. But it can also be too complex, with too many views creating confusion. Again, it does not explicitly address the full impact on the business. In addition, it requires that everyone speak the language, even non-IT professionals.
From this quick summary of the methods, it is apparent that their focus is too ‘short’ – they can be used to produce software quality, but not quality software. The measures also suffer from a similar myopia – McCall’s and Boehm’s models both require large amounts of metrics to be collected, causing a significant administrative overhead to be incurred. At the same time, the full impact of the software on the business is not completely addressed. The models focus on the software and not on the software/business system.

The define-your-own-model would seem ideal, as it allows almost total flexibility. However, the take-up rate is low, although it is highly likely that this method is used informally.

Single aspect measures of quality are similarly flawed. Assessing quality using only a single aspect of quality would seem short-sighted in the extreme. Quality software is the result of a whole host of reasons, as recognised by the models discussed above.

One measure that will capture everything about the software quality is through-life cost. But it is not perfect. It can lack focus, as it will capture all costs, and it may be difficult to disaggregate for analysis. It is also an estimate, and as such, it relies heavily on the assumptions made. Possibly the best measure of quality would be hindsight through-life cost, but this is only available at the end of the lifecycle, when the decisions have already been made!
References:


Accessed 10/02


Walton, M. [1986], The Deming Manufacturing Method, Putnam Publishing Group,
New York, 1986
Appendix III: Review of IT Successes And Failures

Appendix III.1 Introduction

IT systems are rapidly becoming the keystone of business. A successful IT implementation can radically change the business, increasing profit, and enabling new business. Unfortunately, IT failure can destroy businesses, losing the goodwill attached to the brand name, and costing vast sums of money. As will be discussed below, industry IT has a very high failure rate, and scepticism of software engineering techniques is similarly high. It is common practice for industry to present "lessons learned." It seems that it is also common practice to then ignore these lessons!

As presented in the objectives, several IT projects from industry have been selected. These will be analysed, as they present valuable lessons learned. Both successes and failures have been reviewed. It is to be noted that many of the projects cite 'legacy systems' as the reason for initiating the projects.

This appendix explores some of the prevailing perceptions about IT and IS in general. It then proceeds to present some specific case studies, looking for lessons learned that can be applied within the context of the thesis.

Appendix III.2 Business Perception of Information Technology

It is well known and accepted that properly managed IT does have an impact on the bottom line of a business, and chief executives are totally convinced of this fact [Vowler, 2001] [Doran, 2002]. However, there is also considerable scepticism as to the ability for IT to consistently deliver concrete, observable benefit to the business [PA Consulting, 2001] [Butler Group, 2001] [Cushing, 2002a]. McKinsey reported in 2001 [Goodwin, 2001] that spending on IT by US companies had doubled since 1995. The report stated that US companies were spending on average about $3,000 per employee per year on IT. Simultaneously, the US was enjoying incredible economic growth [Goodwin, 2001].
Until very recently, it was held by most that this global was a direct consequence of IT investment by companies [Newing, 2001]. Strassman [1985] has been researching the relationship between strategic variables and profitability since 1982. His recent analysis was based upon the results of 1,585 companies. His conclusion:

“There is no correlation between investment in information technology and profitability. Although some companies achieve spectacular results through their use of IT systems, higher investment per employee does not result in higher return on equity [Strassman, 1985].”

The McKinsey Global Institute echoes this conclusion in another report [Newing, 2001]. The research indicates a shift from unfettered corporate spend on IT to a much more focussed and targeted approach. The conclusion of both reports is that IT spend should focus more on economics and risk, rather than technology. The implication is that IT projects must be chosen carefully and with solid reasoning. Ploughing money into IT has been shown not to generate the desired returns, and thus is not compatible with profit-maximisation. Rather, the route to success is now considered to be the combination of the IT and business functions [Doran, 2002].

The two reports illustrate why it is so vital to have concrete aims and objectives when investing in IT. Merely investing in IT is not sufficient – the investment has to be targeted to exploit opportunity.

**Appendix III.3 Enterprise Resource Planning Software**

Enterprise resource planning (ERP) is notorious for failures, but has also resulted in several notable successes, as covered in 2.3.1.3, 2.3.1.4, and 2.3.1.5. There have been many commentaries on ERP systems in general, [Benesh, 1999] [Conner, 1998] [Davenport, 1998] [Krasner, 2000]. As Davenport [1998] noted:

“ERP systems are expensive and difficult to implement. The growing number of horror stories about failed or out of control projects should certainly give managers pause . . . Some of the blame for such debacles lies with the enormous technical challenges of rolling out enterprise
systems—these systems are profoundly complex... and installing them requires large investments of money, time and expertise... The biggest problems are business problems. Companies fail to reconcile the technological imperatives of the enterprise system with the business needs of the enterprise itself.”

ERP systems are where software development becomes critical. However, as General Motors found [Simons, 2002], it is not just software engineering that must be flawless, but also business process re-engineering. As Conner wryly notes [Conner, 1998]:

“The most obvious problem with ERP implementations is that these projects are so large and so complex that you can’t tell the implementation of the software apart from re-engineering the business”

Benesh [1999] describes five main areas of common management pitfalls resulting in ERP implementation failure. These five areas involve shortcomings in or lack of:

1. Integrated project team planning
2. Managed communications across many people
3. A formal decision-making process
4. An integrated test plan and managed test plan
5. Applying lessons learned from other implementations

The last point is particularly worth noticing. Applying lessons learned from other implementations should also be used across IT, not just specifically for ERP installations. Below is an analysis of several large successes and failures, examining them for generic lessons learned, as recommended by Benesh.

**Appendix III.4 Industry Information System Development**

A survey of IT successes and failures was conducted, in order to derive common success factors and failure modes. Creating tools that seek to avoid the problems that led to failures, and replicate the conditions, as far as possible, that are linked with successes was seen as a very useful exercise.
Appendix III.5 Development Successes

IT successes seem to be rarer than IT failures. It is an accepted fact that a majority of software projects never come to fruition and are never used [Danks, 1997]. Lessons learned from the successes are therefore very valuable.

Appendix III.6 NASA/University of Maryland/CSC

The Software Engineering Laboratory (SEL) was established as a joint venture between NASA/Goddard, the Computer Sciences Corporation (CSC), and the University of Maryland in 1976. The aim of the venture was to:

- Improve the quality of software by reducing defects
- Improve productivity by reducing costs
- Decrease schedule slippage by reducing the average time to develop mission support software

Around 125 NASA projects were analysed by applying, measuring, and analysing the effect of software process changes. The result was an adoption of an empirically determined set of process changes aimed towards improving software development. The outcomes were a 75% reduction in defects, 59% reduction in costs, and a 25% reduction in cycle time [Basili et al, 1995].

The SEL experience, although concentrated solely on the development, shows the impact that a rigorous process for achieving quality can have a significant impact, with projects running both to time and budget. The empirically-based approach obviously yielded significant results. The approach adopted by the CSC also required working out what to measure, and how to measure it.

Appendix III.7 International Business Systems (IBM)

IBM began a process improvement initiative in 1989, based at the Santa Teresa Laboratory. Starting from an already high base, the products from the Laboratory already had fewer defects than the industry average, the process improvement initiative yielded startling results. The average number of defects dropped by 46%, service costs dropped by 20%, revenue per employee rose 58%, and customer
satisfaction increased by 14% [Kaplan et al, 1995].

IBM has become one of the world's premier IT consultancies. Central to IBM's success in this arena has been a concentration on the service aspect of consultancy. Reducing defects and costs, and increasing customer satisfaction are critically important to maintaining IBM’s reputation. Again, the concentration is on the development of the software, ignoring the long term maintenance aspect. However, it corroborates the SEL’s experience that strong process management gives rise to software development successes. The key to the management seems to be the setting of appropriate measures, and monitoring these measures closely. It could be suggested that the reduction in pre-release defects also reduces the maintenance burden. However, it does depend upon which defects are detected and removed. Those with a low probability of causing an operational failure would not significantly reduce the maintenance burden.

**Appendix III.8 AT&T Bell Laboratories**

In 1990, AT&T Bell Laboratories aimed to triple software productivity within three years. Alongside this dramatic increase in productivity, it was planned to increase software quality as well. The main thrust of the reduction was in decreasing complexity of programs. It was hoped that the reduction in complexity would make coding quicker, and also assist quality assurance. The increase in quality would create a positive feedback effect, as there would be a reduced need for rework. Two years after the introduction of changes affecting product administration, software configuration management, component reuse, languages, and improved project management, productivity had almost tripled [Belanger et al, 1990].

The AT&T experience shows the value of setting stretch targets. Though the stated aim of tripling the software productivity seemed ambitious, the target was achieved. AT&T approached the issue at a different angle to both the SEL and IBM. AT&T concentrated on complexity, an aspect that contributes significantly to through-life maintenance costs. This is a narrower focus on one aspect of the software than IBM and the SEL. However, the solutions adopted were of a much broader focus – involving both software improvements, such as the component reuse, and human-
aspect improvement, changing the processes itself. The holistic approach to the central problem was obviously beneficial.

Appendix III.9 Pilkington Glass

A series of acquisitions had left Pilkington, a European glass manufacturer, with a very fragmented and haphazard IT architecture, consisting of many different legacy systems. Seeking efficiency gains, Pilkington launched an ambitious project to integrate as many of these legacy systems as possible [Cushing, 2002b]. After an extensive change-management exercise, Pilkington chose SAP R/3 to centralise planning and scheduling of float glass production throughout Europe. Pilkington would replace the various legacy IT systems with SAP modules covering finance, fixed assets, purchasing, inventory management, warehouse management, and production planning.

The company made a point of utilising both SAP consultants and internal business staff to ensure business ownership of the implementation. The team then drove through the initial planning, corresponding with all the major stakeholders across Europe. Communication between business units, and different levels of management, was seen as being critical.

The new system was rolled out on a country by country basis, over a two-year period. This model allowed Pilkington to modify the installation to allow for local requirements. However, this flexibility was offset by the fact that the modifications had to flow back into previous implementations, making each succeeding roll-out more time-consuming.

The implementation involved significant process change, which caused some difficulties, according to Phil Roberts (the Pilkington IT infrastructure manager). The implementation cost Pilkington £21m. However, the company expects payback to occur in 2003, due to the operational cost savings [Cushing, 2002b].

The Pilkington experience shows the value of communication and corporate involvement, in particular when creating an enterprise-wide strategy for resolving a
Appendix III

legacy system problem. The implementation team made a point of analysing the problem space, and eliciting user opinion at all stages of the project. The communication also extended to the implementation phase, where the flexible approach made more work for the implementation team, but significantly contributed to the success of the project, as the business units obviously felt a sense of ownership of the implementation. The importance of senior management support of the project is clear, along with the necessity of clear communication between users, developers, and management. The communication was obviously two-way, and conveyed both aims and methods clearly.

Appendix III.10 Anglian Water

In 1996, during its Y2K-compliance exercise, Anglian Water recognised that it had a significant legacy system problem. The company was endeavouring to move away from its existing collection of heavily customised legacy systems towards Y2K-compliant client-server systems. The current legacy systems were all business-critical systems, passing data across hand-coded interfaces [Vowler, 2000].

At the time, Anglian Water operated four principle production systems:

- SAP R/3 - running on MS Windows NT and SQL/Server
- A UNIX-based jobs and asset-management, and job scheduling system
- New Billing System, a bespoke system written in the 4GL Pacbase, running on the GCOS 8 mainframe and the Codasyl database
- Operational Common Database (addressing module) - a bespoke legacy system running under Digital Equipment VMS and RDB on a Vax platform

The challenge facing Anglian Water was to integrate these four, disparate systems together by October 1998 (the go-live date for compliant systems). There were in existence over 20 data conversions and 12 data interfaces. The issue was further complicated by the need for the interfaces to be written in COBOL.

Due to the lack of experienced, available COBOL programmers, Anglian Water decided to use enterprise application integration tools that would automate the production of interface code. There was a product evaluation, the outcome of which
was the decision to use ETI’s Extract. This produced not only well-documented COBOL code, but it was also capable of producing ABAP (SAP-specific code) code for use in an SAP system.

Anglian Water also trained non-IT people to use the Extract software, on the basis that the business users would understand the data, and be able to apply business rules more quickly than an IT coder, who would have to define the business rules from first principles. By the end of the project, which was delivered ahead of schedule, Extract had generated over 450,000 lines of code, in roughly a quarter of the time required by the traditional, hand-coding, methods.

Key success factors for Anglian Water’s success were its willingness to examine and use non-traditional methods, and its clear definition of its problem space and what the solution would have to be able to do in order to solve the problem. Communication between the various stakeholders was also vital, as the project itself carried a significant amount of risk. The project would have failed without senior management support [Vowler, 2000].

**Appendix III.11 Sandia National Laboratories Configuration Management System**

As reported by Bray and Hess [1999], developers at Sandia National Laboratories have been maintaining and evolving a configuration-management system to support and manage complex and extensive engineering documentation for drawings of high-tech weapons systems developed for the US Department of Energy and Defense for a number of years.

The original legacy system that was replaced towards the end of the 1990s was a specialised batch-oriented file system, consisting of over 200 separate programs. The system supported five master files and around 5.7 million records. The purpose of any configuration management system is to let engineers include administrative information on engineering drawings, report on the configuration of assemblies and components, and track both the release and change history of parts.
The original system at Sandia was designed and built on a Unisys mainframe. Maintenance costs of this legacy system had reached over $1 million a year. The decision was made to re-engineer the whole system. This was done by:

- Installing local area networks
- Creating a database server to house the administrative data drawings and process data requests in SQL against the RDBMS
- Creating application servers to execute the Windows application interface the customer uses for on-line, real-time access to the information.
- Creating application clients that simply display the Windows application to the customer

The work was constrained in that whatever system was designed, it had to be UNIX (SPARC workstations) and PC compatible. The re-engineering was both simplified and made more complex by the fact that Sandia did not want to rewrite the underlying business processes, just the system itself.

The solution required the use of a combination of natural language information modelling, design recovery, and procedural re-engineering to gain knowledge about the system. There was no change to the original functionality.

The re-engineering group of six people had a $500,000 budget, and a requirement that the project pay for itself within one year of project completion. As it turned out, all objectives were met, with the exception of the payback period, which turned out to be 18 months.

The developers attributed this remarkable success to [Bray and Hess, 1999]:

- Use of software experts
- Data and business system experts
- Solicited feedback from in-house experts on the business, data and the system
- Software engineers with varying areas of expertise (reading source code, interpret software constituents, transactions, and business rules) with an
ability to implement reengineered software constructs in the new environment

• Automated tools, some of which were COTS, and some were bespoke
• Used vendor tools, and created custom tools

The Sandia experience clearly shows how IT success is reliant upon good communication and adequate planning. Users were heavily involved in the design of the system, and frequent referrals to experts were common. The development team had a clear plan to work to, but had flexibility within that plan to do what was needed. One of the success factors identified by the development team was the use of bespoke tools. This shows that the team understood the limitations of COTS products and used them only where appropriate. The development also made extensive use of expert knowledge and judgement.

Appendix III.12 Development Failures

IT failures are reported in the press every month. ERP systems seem particularly prone to producing failures. Just as lessons can be learned from successes, so too from failures.

Appendix III.13 Atlanta Olympic Games/IBM

The Olympic Games in Atlanta, 1996, was a major international event. IBM was contracted to handle the information technology that would support the Games. To this end, IBM installed four S/390 servers, 16 RS/6000 servers, and 80 AS/400s [Danks, 1997]. There were 7,000 PCs, 1,000 desktop printers, all interconnected on 300 local area networks. Further integration occurred, pulling 13,000 telephones, 11,500 televisions, 6,000 pagers, 9,500 mobile radios, 80,000 cable installations, and 35 accreditation stations. Adding to the complexity, ten different companies supplied 450 security access points, including Swatch (timing), Xerox (document reproduction), Bell South (networking), AT&T (telecommunications), Motorola (paging), Panasonic (monitors), and Kodak (image processing).

The project was a disaster. The data services were so slow that information on the first day of the Games was not able to make the American Sunday papers the next
day. Programming errors on the News feed and Olympic WWW system rendered the local information kiosks useless. The information system designed to aid commentators was also misleading. It was the latter that was most publicised – the system insisted that a gymnast was 97 years old and an Angolan basketball player was 3 feet tall [Danks, 1997]. Testing of the network was cut short, primarily for cost reasons.

IBM was chosen because of its reputation for quality. As shown above, IBM has already had some considerable success with software development. However the Olympic Games was not really a software development project, but rather a hugely complex integration project. Risks were seemingly incurred with the inadequate requirements capture and the lack of testing. Many of the embarrassing errors could perhaps have been avoided by a rigorous testing methodology.

**Appendix III.14 Nike**

Order-management problems cost Nike Inc. $80 million to $100 million in sales for the third quarter of 2000. Nike issued a warning that blamed its new $400 million demand and inventory management system for an expected earnings shortfall of up to 21 cents per share. Nike implemented this new system in an effort to remove the many legacy systems that it previously operated, and replace it with a single, centrally maintained system. After rolling out the system, orders for some shoes wound up being placed twice – once by the new system and once by a legacy order-management system, or else not at all [Konicki, 2001].

Nike was criticised by i2, the vendors of the chosen IT solution, for failing to follow i2's recommendation to minimise customisation, to use its best practices for the footwear and apparel business, and to deploy the system in stages [Konicki, 2001].

The Nike experience shows that requirements specification is vital, and that thought must be given to the whole of the development, including the last stage, that of rolling out the finished product. In hindsight, Nike's decision to roll-out the application to everywhere at once, the so-called 'big bang' approach, was the wrong one to have adopted, but it would be impossible to say whether i2's staged approach would have...
worked any better. It also shows the perils of over-customisation of COTS products.

Appendix III.15 Department 56

In January 1999, a giftware and collectibles company, Department 56, decided to buy enterprise-wide operating systems that would improve productivity and give it greater product order and fulfilment visibility. The intention was to replace existing legacy systems with a single, enterprise-wide system. The system it purchased was defective, according to the company, and it is suing Arthur Andersen Worldwide (now Accenture) for damages.

The inadequate implementation crippled Department 56's ability to "fulfil nearly every mission-critical operation in its business," according to Chairwoman and CEO Susan Engel [Cottrill, 2001].

The project cost the company about $8 million in revenue. The company took a combined $18 million charge attributable to accounts receivable and customer claims resolution stemming from system failure. Orders got lost as the information that was supposed to be exchanged between handheld units and PCs went astray. The fulfilment process was prone to failures. Some orders were not filled accurately or the shipping information did not tally with what the customer has requested. One of the most damaging problems was that the system was not able to invoice concurrently with the shipment.

Another major problem area was the inventory and warehouse module. The primary software components were a product of J.D. Edwards called OneWorld. The inventory management and warehouse module, an older version J.D. Edwards World product, was never stated to be compatible with the newer OneWorld element of the ERP [Cottrill, 2001].

As in the Nike fiasco, Department 56/Arthur Andersen neglected to fully scope the requirements for the system, and the consequences to the business of not having defined capabilities. The incompatibility issues between components should have been caught at a much earlier stage – compatibility between modules should have
been an identified requirement.

Appendix III.16 Federal Aviation Authority

The US Federal Aviation Administration Advanced Automation System was supposed to replace an existing legacy air traffic control system in the US. The replacement system was scoped in 1981. The Advanced Automation System project was begun in 1984. The Federal Aviation Administration granted an Advanced Automation System contract for $4.3bn. The Government Accounting Office stated, in 1992, that continuing delays in the Initial Service Sector Suite, a key component of the Advanced Automation System, could “have the potential for affecting Federal Aviation Administration’s ability to handle safely the predicted increases in traffic into the next century [Carleton, 1998].” A 19-month delay to rollout of Initial Service Sector Suite was blamed on underestimation of testing time. A further 14-month delay announced in 1993 was also blamed on Initial Service Sector Suite-related problems. In 1994, a 31-month schedule delay was estimated, alongside an increase in the estimated cost for the deliverable – now standing at $7bn. The Federal Aviation Administration then suspended the Advanced Automation System programme, replacing it with a scaled down programme called the Display System Replacement. The replacement system was finally delivered 17 years after it was defined, at a cost of $5.6bn [Carleton, 1998] [Lopez, 1994].

The Federal Aviation Administration have since announced a $1bn air traffic modernisation contract, termed STARS, to make up the shortfall behind the scaled-down display system replacement programme. The main thrust of this improvement program was to replace the 20 year-old equipment that had been scheduled to be replaced by the Advanced Automation System, but had not been replaced by the display system replacement programme. The contract was awarded to Raytheon in 1996 [Pena, 1996]. Unfortunately, STARS is suffering from the same malaise that afflicted the Advanced Automation System program. The Federal Aviation Administration was forced to inject an additional $462m in 1999, above the $940m already agreed. Completion is expected in 2008, four years behind schedule, but even that estimate is considered hopeful [Anon, 2001].
For both Advanced Automation System and STARS, it seems that the Federal Aviation Administration did not have a set of requirements that the developers could work to. Federal Aviation Administration acting administrator Joseph Del Balzo admitted as much in an interview with Flight International [Anon, 1993]. By appearances, it seems that the developers were not able to explore the full functionality required by the Federal Aviation Administration. The Federal Aviation Administration, for its part, consistently underestimated the complexity and scale of the problem. The projects seem to have suffered from significant scope creep and management uncertainty as to what to do with the project at various stages. As Clemens notes [Beechener, 1999], understanding the problem, and the complexity of the situation is critical to project success, and this was not done in by the Federal Aviation Authority. The management seemed to have been firmly behind the idea of modernisation, but with little idea of what, how, and when to modernise. The targets that were set seemed to be unobtainable, and the combination of all these factors made both projects run significantly over-budget and behind schedule.

Appendix III.17 Internal Revenue Service

A Computerworld investigation found that delays in overhauling the federal tax systems were costing the US Treasury as much as $50bn a year [Anthes, 1996]. Representative Jim Lightfoot said that “The IRS (Internal Revenue Service) has spent $4bn on TSM [Tax Systems Modernization] and has nothing to show for it.” The Computerworld investigation reported that the Internal Revenue Service had:

- Failed to perform business process redesign before it began the systems development
- Neglected to develop an overall systems architecture or development blueprint
- Employed primitive, and at times, chaotic software development methodologies
- Failed to manage information systems as investments
- Neglected information security

In December 1995, the National Research Council found “serious deficiencies” in the Internal Revenue Service technical management, systems architecture, process improvement, and systems security. A later Government Audit Office report stated that the Internal Revenue Service has “provided little tangible evidence that actions
being taken will correct the pervasive management and technical weakness that place TSM, and the huge investment it represents, at risk. [Anthes, 1996]"

In this IT failure, there are myriad causes. As highlighted above, communication and adequate planning and foresight were neglected. Bad money followed good, as there was neither an ability to stop the project, nor an willingness by senior personnel to admit failure. The requirements specification was obviously incomplete, and product development was affected by both the lack of a concrete plan to work to, and a continually changing set of business rules. Also shown is the necessity to look at IT merely as part of the whole business process – a holistic approach is obviously more beneficial than the approach undertaken by the Internal Revenue Service.

Appendix III.18 National Air Traffic Services Centre

The National Air Traffic Services control centre at Swanwick has been a high profile IT failure. The intent was to replace the aging legacy air traffic control systems covering the majority of southern UK airspace. The project was originally intended to go live in 1997, but significant problems were disclosed when the product, provided by Lockheed Martin, was tested. As a result, the project eventually went live six years late and £150m over budget [Ranger, 2002] [Fallis, 2002].

Evidence has shown that senior officials, including the Transport Secretary Stephen Byers, systematically ignored expert advice [Clement, 2002], and did not exercise good control over the project, which suffered a series of 'scope creeps' and periodic uncertainty over funding levels.

A National Air Traffic Services spokesperson admitted that the "original timetable was optimistic" [Ranger, 2002]. National Air Traffic Services is already looking for a replacement system, as the delay has brought the system to near obsolescence. COTS systems are the preferred option at this stage.

The business case was also optimistic. The original business case for the centre assumed a certain level of flights. Subsequent to the global slowdown on flights following the September 11 disasters, the Swanwick centre became increasingly
uneconomical [Doran, 2002]. The Government was forced to inject more money into the semi-privatised business [Clement, 2002] in January 2001, and yet another bridging loan, worth over £60m, was provided by banks and the Government in the first quarter of 2002 [Done, 2002].

There seems to have been many causes for failure in this case. Firstly, the scale of the task at hand was underestimated, and possibly not fully evaluated. Scope creep obviously occurred, resulting in the schedule and the budget being significantly overrun. All the options were not considered. A NATS spokesperson admitted that NATS has been forced to re-examine the whole way it approaches these kinds of project [Ranger, 2002]. NATS is now considering a move towards COTS products and away from large, bespoke systems [Beechener, 1999].

**Appendix III.19 Industry Experience of Large IT Systems - Summary**

As reported by Danks [1997], the Standish Group has found that more than 84% of US IT projects fail to meet original expectations, and 94% of those have to be restarted. This is not solely a US problem either. Coopers & Lybrand’s Computer Assurance Services risk management group revealed that, in a survey of 80 large enterprises in the UK including 22 of the FTSE-100 companies, 85% reported IT projects that had run significantly over time, over budget, and had not delivered expected benefits [Danks, 1997]. Gartner estimate that firms waste $500bn every year on ill-conceived IT projects [Huber, 2002]. Jones [1995] stated that the failure and cancellation rate of large software systems ran at over 20% in 1995. Of the 80% of projects that were completed, approximately two thirds were late and over-budget, some more than doubling their estimated cost. About 60% of the projects also had reliability and quality problems.

Against this background, it is obvious why the lessons from successful projects must be duplicated. Understanding what makes successes and failures is critical to providing profit maximising opportunities for industry. All the successes and failures have recurring themes. This is the philosophy underlying the CMM concept. The successes described above have carefully defined requirements, acceptable to managers, users and developers. There is a clear link to an overall strategy, either in
the form of a cohesive IT strategy, or a business strategy. Achievable targets are set, with no hopeful estimations of what could happen. There are clear lines of communication, between users, developers and managers. As described by McRay and Hancock [1995], there must be a clear route for decision-making. The conclusion from the examples above that senior management support is crucial, Vowler [2001, 2002] mentions the requirement for senior management support, and that they have buy-in for both the problem and solution. King [2002] also highlights the critical nature of senior management support.

Soft factors, such as politics, project management, and people management have been reported by Schneider [2002] as being critical for IT project success. In a survey of 900 senior IT professionals, Computer Weekly and The Coverdale Organisation found that good communication and project management are key to successful IT projects [Schneider, 2002]. This finding is replicated in the examples above.

IT failures seem to lack some or all of the characteristics that make a successful project. There have been many examples of poor requirements engineering or lack of management support, as mentioned above. This means the problem has not been clearly defined, and consequently the solution is equally ill-defined.

One conclusion that is inescapable, as Bentley [2002] highlights, is that the IT needs to be driven by the business problems within a highly specific context. This point is driven home by a number of commentators [Chandrasekhar, 2002] [Koch, 2001] [Manchester, 2000]. Every company faces a vast array of different variables that will affect the success or failure of any project. Due to the large number of variables, a generic model intended to replicate software successes cannot succeed. Any model that is generic enough to be moved outside of where it was developed is most likely going to be too generic to be used without customisation. Methods and tools that merely assist in defining the business problems are transportable however. Methods and tools that are configurable to specific situations, but produce output in a standard presentation would be useful. Standard presentations will assist in improving communication, and most importantly, in gaining senior management support for IT projects.
To summarise, identification and management of stakeholders is vital. Soft factors, such as politics, communication routes, people and project management are vital. Communication between stakeholders (including senior management, developers, and users) is critical. A clear problem definition is necessary. To define the problem, it is necessary to have a basis for reference. When a business says “We have a problem with...” it is important to be able to define precisely what the problem is. Terms such as quality and requirements must be adequately defined.

Many of the examples detailed above were precipitated by the need to resolve ‘legacy system’ problems, such as:

- Old/obsolete hardware
- Old/obsolete languages
- Upgrade issues
- A desire to future-proof an IT system

The pace of evolution of IT systems is staggering. Hardware and software can become obsolete within a few years. Obviously companies cannot run obsolete equipment without incurring significant maintenance costs, so the replace strategy is often used on what are termed ‘legacy systems.’ Defining legacy data seems straightforward, but the term ‘legacy system’ is bandied about, and is often cited as the reason for replacement and/or modernisation. When does a system become legacy, and what is legacy? Chapter two explores how various people have defined legacy systems, and why they are considered problematic.
Appendix III

References:

Beechener, J. [1999], “Procurement strategy holds the key to project success,” *Jane’s Airport Review*, 01/03/99, pg. 14
Cottrill, K. [2001], “Retailer Sues Arthur Andersen,” *Traffic World Magazine*, 12/03/01
Cushing, K. [2002b], “Stem the flood of IT waste,” *Computer Weekly*, 18/07/02, pg.22
Done, K. [2002], “Air traffic controllers seek talks on Nats cash crisis”, *Financial Times*, 05/08/02, pg. 2
Doran, C. [2002] “Change your view of IT,” *Computer Weekly*, 01/08/02, pg. 28
Doran, J. [2002], “Finance Backers Threaten to Pull Out of NATS Rescue”, *Times*, 28/01/2002, pg. 21
Huber, N. [2002], “Gartner: firms waste £351bn each year on ill-conceived IT projects,” *Computer Weekly*, 21/03/02
King, J. [2002] “Make IT bigger, better, faster,” *Computerworld*, 01/01/02
Koch, M. [2001] “Leverage legacy systems with a blend of XML, XSL, and Java,” *ITworld.com*, 26/01/01
Konicki, S. [2001], “Nike just didn’t do it right,” *Informationweek*, 05/03/01
McRae, J, Hancock, C. [2002] “Accessing the Corridors of Power”, *Guardian*, 07/02/02, pg. 10
Newing, R. [2001] “Watch the economics and the risk, not the technology,” *FT IT Supplement*, December 5 2001, p. VIII
Schneider, K. [2002] “Non-technical factors are key to project success,” *Computer Weekly*, 28/02/02, p. 4
Simons, M. [2002] “GM rebuilds SAP system,” *Computer Weekly*, pg. 6, 14/02/02
Appendix IV: Software Metrics

Appendix IV.1 Introduction
This appendix covers derivation of some of the complexity metrics considered, such as McCabe’s cyclomatic complexity and various coupling and cohesion metrics. It is derived linear regression terms, which were used in the first, pilot stage, and subsequently discarded.

Appendix IV.2 Types of Complexity Metrics
Complexity can be considered as another, single aspect of quality. Code that is unnecessarily complex can be considered to be of poor quality. Several measures of complexity were considered for use in the case study.

Appendix IV.2.1 McCabe’s Cyclomatic Complexity
McCabe proposed that program complexity could be measured by the cyclomatic number of the program’s flowgraph [McCabe, 1976]. For a flowgraph $F$, McCabe’s cyclomatic complexity metric is given as (from [McCabe, 1976]):
\[ v(F) = e - n + 2 \]  
Equation A.1
where $F$ has $e$ arcs and $n$ nodes. McCabe’s metric is one of the most referenced and well-known metrics. However, it can be misleading. Fenton and Pfleeger [1997] show that the cyclomatic complexity metric may not paint a complete or accurate picture of program complexity. McCabe also proposed a second measure of the overall structuredness of a program with a derivative of the cyclomatic number.

Appendix IV.2.2 Halstead’s Complexity Metric
The second complex complexity metric examined was Halstead’s complexity metric [Halstead, 1977]. Halstead defined a program $P$ as a collection of tokens. A token can be either an operand or an operator. Each token has the following measures:
\[ \mu_1 = \text{number of unique operators} \]
\[ \mu_2 = \text{number of unique operands} \]
$N_1 = \text{frequency of operators}$
$N_2 = \text{frequency of operands}$
The length of $P$ is given by $N$ in equation A.2:
\[ N = N_1 + N_2 \]  
Equation A.2

The vocabulary of $P$ is given by $\mu$ in equation A.3:
\[ \mu = \mu_1 + \mu_2 \]  
Equation A.3

The volume, $V$, of $P$ is proxied by the number of mental comparisons needed to write a program of length $N$ in equation A.4:
\[ V = N \times \log_2 \mu \]  
Equation A.4

The program level of program $P$ of volume $V$ is given by equation A.5:
\[ L = \frac{V^*}{V} \]  
Equation A.5

where $V^*$ is the potential volume. The potential volume is the minimum size implementation of program $P$. The inverse of the program level is defined as the difficulty, $D$ equation A.6:
\[ D = \frac{1}{L} \]  
Equation A.6

Halstead then derives an estimate of the program level, $\hat{L}$ in equation A.7:
\[ \hat{L} = \frac{1}{D} = \frac{2}{\mu_1} \times \frac{\mu_2}{N_2} \]  
Equation A.7

Estimated program length is given by equation A.8:
\[ \hat{N} = \mu_1 \times \log_2 \mu_1 + \mu_2 \times \log_2 \mu_2 \]  
Equation A.8

Estimating the complexity of any given program is a function of its level, as indicated above.

Both McCabe's cyclomatic complexity and Halstead's effort metric have been criticised on many grounds. Both metrics have been criticised on the underlying psychological model [Coulter, 1983]. The empirical support Halstead's metric has received has been attacked in Hamer and Frewin [1982]. Difficulties with Halstead's counting rules have been documented by Lassez et al [1981]. Shepperd [1988] notes a severe difficulty in assessing McCabe's metric due to the lack of an implicit model. He also draws a distinction between intra- and inter-modular complexity. He recognises that the metric has not been applied in a standard way, often intra-modularly in one study, and inter-modularly in another. He states that the derivation
of \(v(G)\) makes inter-modular comparisons rather suspect. However, the work of Basili and Perricone [1984] cast doubt on the use of this metric as an intra-modular comparator.

\(v(G)\) had several disadvantages from the point of view of the project. Firstly, it required a detailed knowledge of the program code itself. Secondly it also required detailed knowledge of the language the code was written in, so as to create the program control graph.

A final metric examined was developed by Oviedo in his study of program complexity in the context of high level languages [Oviedo, 1980]. As for McCabe’s \(v(G)\), this required logical decomposition of the program to separate nodes. Production of a program control graph then allows examination of the properties of the abstraction. The program complexity model as developed by Oviedo, shows that program complexity, \(C\), is given in equation A.9:

\[
C = CF + DF
\]

where \(CF\) is the control flow complexity, and \(DF\) is the data flow complexity.

**Appendix IV.2.3 Coupling and Cohesion**

Coupling is defined as being the degree of interdependence between modules [Yourdon and Constantine, 1979]. It is usually used hierarchically, with coupling between pairs of modules being used to derive a global coupling metric [Fenton and Pfleeger, 1997]. There is no standard measure of coupling. However, Fenton and Pfleeger [1997] have created an ordinal classification for coupling, based upon the type of relationship that exists between any two given modules. Fenton and Melton [1990] have developed an ordinal scale measure of coupling.

Troy and Zweben [1990] identified a range of different counts that can be derived from a structure chart that each contributes to coupling. The measures identified include:

1. The maximum number of interconnections per module
2. The average number of interconnections per module
3. The total number of interconnections per module
4. The number of modules accessing control interconnections
5. The number of data structure interconnections to the top-level module

Troy and Zweben hypothesised that coupling was one of the most important attributes affecting the quality of software design. However, without specific measures of coupling, they were unable to formally prove their hypothesis.

Cohesion is similar to coupling, in that where cohesion measures how dependent modules are on other modules, coupling measures how many modules do similar actions. It was also proposed by Yourdon and Constantine [1979]. Cohesion is the extent to which the individual components of a module are need to perform the same task. Again, there are no standard measures of cohesion, but Yourdon and Constantine did propose classes of cohesion, giving an ordinal scale of measurement [Yourdon and Constantine, 1979]. It is often twinned with coupling, since conventional wisdom has it that modules with high cohesion and low coupling are desirable [Fenton and Pfleeger, 1997]. Macro and Buxton [1987] added the concept of functional and abstract cohesion, reflecting the need to modify the definition of cohesion to reflect the modern capability to encapsulate abstract data types as modules. A cohesion ratio can be defined in equation A.10:

\[
\text{Cohesion ratio} = \frac{\text{number of modules having functional cohesion}}{\text{total number of modules}}
\]

Equation A.10
(from [Fenton and Pfleeger, 1997])

**Appendix IV.2.4 Function Points**

There have been many attempts to move away from the inefficient and error-fraught lines of code measure. Albrecht's function points constitute one of the more influential measures. Function points are intended to measure the amount of functionality in a system as described by a specification [Albrecht, 1979].

Function points are derived by first computing the unadjusted function point count, by determining the number of items of external inputs and outputs, external inquiries, external files, and internal files.
The unadjusted function points are then assigned a subjective complexity weighting on a three-point ordinal scale. The count is derived by summing the total function point count and then multiplying the number of items by the weighting. Finally the unadjusted function point count is multiplied by a technical complexity factor. Fenton and Pfleeger [1997] illustrate the derivation and calculation of function points.

Albrecht's reasons for proposing the function points as a measure of system size are [Albrecht, 1979]:

- The measure isolates the intrinsic size of the system from the environmental factors, facilitating the study of factors that influence productivity
- The measure is based on the user's external view of the system, and is technology-independent
- The measure can be determined early in the development cycle, which enables function points to be used in the estimation process
- Function points can be understood and evaluated by nontechnical users

The size of the task of designing and developing a business computerised information system is determined by 3 factors:

- Information processing size
- Technical complexity factor
- Environmental factors

Function point analysis is a method used to determine the relative size of a system based on the first two factors. The information processing size is determined by firstly identifying the system components as seen by the end-user, and then classifying them as one of 5 types

- External (logical) inputs
- Outputs
- Inquiries
- External interfaces to other systems
- Logical internal files

Individual components of the program can be further classified as:
Depending upon the number of data elements in each type (and other factors), each component is then given a number of points depending on its type and complexity.

II.2.4.1 Critical Review of Albrecht’s Function Points

The classification of all system types as ‘simple’, ‘complex’ etc, has the merit of being straightforward, but seems to be rather oversimplified. For example, a system component with 100 data elements is given at most twice the points of a component with one data element.

The choice of ‘weights’ reflecting the “relative value of the function to the user/customer” was determined by Albrecht “by debate and trial”. Symons [1988] thought that a more objective approach might yield better results. He notes that surprising weightings have evolved: inquiry via a batch input/output combinations gain twice as many points as the same inquiry provided on-line. The degrees of influence (weights) are restricted to the 0-5 range, which may not always be satisfactory.

II.2.4.2 Mark II Function Points

Symons [1988] advanced the use of function points with a few amendments, starting by addressing basic assumptions:
1. Regard a system as consisting of logical “transaction types” (logical input/process/output combinations).
2. Interfaces at this logical level will be treated as any other input
3. Inquiries will be considered just as any other input/process/output combination
4. The concept of a “logical file” is almost impossible to define unambiguously, particularly in a database environment.
5. At this level, the concept of “file” is not appropriate
System size scale will be related explicitly to the effort to analyse, design and develop the functions of the system. This assumption contrasts with Albrecht’s aim of having a size which represents the ‘value’ of function delivered to the user.

Symons [1988] suggested having McCabe’s process complexity measure as a proxy for the process component. This size measure of the process component is validated with the assumption that a well-structured function should match the logical data structure. Another measure of processing complexity is to count the number of entity-types referenced by the transaction-type, although this seems a rather crude, first-order method.

The result of these new assumptions is given by equation A.11:

\[ \text{UFPs} = N_i W_i + N_e W_e + N_o W_o \]  

where:

- \( N_i \) = number of input data element types
- \( W_i \) = weight of an input data element type
- \( N_e \) = number of entity-type references
- \( W_e \) = weight of entity-type references
- \( N_o \) = number of output data element types
- \( W_o \) = weight of an output data element type
- UFPs = Unadjusted function points

Equation A.12 gives the Mark II function point count, derived from equation A.11:

\[ \sum_{i=1}^{n} (N_i + N_e + N_o) \]  

summed over all transactions

There is a general tendency for the Mark II method to give larger information processing size relative to Albrecht’s, as system size increases. There is also the tendency of the Mark II method to show a high degree of sensitivity, especially in
smaller systems, to relatively high or low average numbers of data elements per input or output component.

Symons himself notes that function points are not a technology-independent measure of system size [Symons, 1988]. The technology dependence is implicit in the weightings used.

Symons [1988] also mentions that if new technology is employed, the weightings must be re-estimated. If an organisation changes its whole technology, and it wants to continue to make a fair size and productivity comparisons, it must calibrate a new set of normative function point weights.

Appendix IV.3 Use of Regression Analysis of Reported Software Defects

Regression analysis is employed as a simple method of determining whether a linear relationship exists between a dependent variable and n independent or explanatory variables. For notational purposes, Y represents the dependent variable, and X the independent variable. If there is more than one independent variable, then the various independent variables will be shown by an associated subscript, for example, X_1, X_2, ..., X_n. One drawback of regression analysis is that a relationship does not imply causality.

Appendix IV.3.1 Regression Analysis Methodology

Linear regression analysis attempts to fit a line through the series of data points. The line is described by an equation. The general regression equation gives the line precisely as:

\[ Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i \]

where \( \beta \) represents the coefficients, and \( \varepsilon \), the error term. Equation A.13 describes where any point \( i \) is likely to be in XY space. The model, as presented in equation A.13, can be divided into the deterministic, or non-stochastic, part and the stochastic part, which is represented by the error term.
The stochastic nature of the error term entails that all variation is due to random effects – it cannot be predicted. Conversely, non-stochastic variables can be predicted, hence their alternative nomenclature – deterministic variables.

Appendix IV.3.2 The Sample Regression Function

Equation A.13 assumes that the model can be applied using data from the entire population – it is the population regression function. Unfortunately, it is rare to work with the entire population, therefore the population must be sampled, and then an estimate of the population regression function is made, using the data gathered from the sample. This results in the ‘fitted line’ as described by equation A.14:

\[ Y_i = \hat{\beta}_0 + \hat{\beta}_1 X_i + e_i \]  

Equation A.14

Appendix IV.3.3 ‘Good’ Estimators

The Gauss-Markov theorem states that if certain assumptions are made, all estimators will be best linear unbiased estimators [Lehman, 1951]. If there is a significant chance that any of the assumptions below are not correct, the regression model will not be an accurate portrayal of the system it is attempting to model. The assumptions are:

1. The true model is linear and correctly specified by the regression equation.
2. The independent variables are nonstochastic
   \[ \rho(X, e) = 0 \]  
   Equation A.15
3. The error term has a zero mean
   \[ E(e) = 0 \]  
   Equation A.16
4. The error terms are non-correlated – no auto- or serial correlation
   \[ \rho(e_i, e_j) = 0, \ \forall i \neq j \]  
   Equation A.17
5. The error terms have a constant variance – homoscedasticity
   \[ \sigma_i^2 = \sigma_j^2, \ \forall i, j \]  
   Equation A.18
Appendix IV.4 Fitting the Line

To fit the best line, it is necessary to minimise the distance between the actual data points and those on the fitted line. This was accomplished using the ordinary least squares technique (OLS), which chooses the estimators, $\hat{\beta}_0$ and $\hat{\beta}_1$, such that the residual $e_i$ is as small as possible. This is achieved by minimising the residual sum of squares:

$$\text{Minimise } \sum e_i^2 = \sum (y_i - \hat{y}_i)^2$$  

Equation A.19

Appendix IV.5 Goodness of Fit

The adequacy of the fitted line is initially given by the coefficient of determination, termed $R^2$. It represents the proportion of variation in the dependent variable that has been explained or accounted for by the regression line. The values of $R^2$ range from zero to one. A value of zero indicates that none of the variation in $Y$ is explained by the regression equation; a value of one indicates that all of the variation in $Y$ is explained by the regression equation.

$R^2$ is calculated by using:

$$R^2 = \frac{\left(\sum xy - \frac{n\bar{x}\bar{y}}{n}\right)^2}{\left(\sum x^2 - \frac{\bar{x}\sum x}{n}\right)\left(\sum y^2 - \frac{\bar{y}\sum y}{n}\right)}$$  

Equation A.20

A general rule of thumb is that an acceptable $R^2$ value is greater than 0.25. Anything less than 0.25, it is considered that the explanatory power is due to random variation, not the underlying correctness of the theory that was used to derive the model.

Appendix IV.5.1 Hypothesis Testing

Two hypothesis tests are employed – the t-test and the general F test. These will be described below:

Several t-tests will be used. It can be shown to be distributed:

$$\frac{\bar{X} - \mu}{S/\sqrt{n}} \sim t_{(n-1)}$$  

Equation A.21
Where $\bar{X}$ is the mean of sample, $\mu_x$ is the population mean, $S$ is the square root of the sample variance, and $n$ is the number of observations.

The F test tests the significance of the overall regression. This is given by:

$$F_{1,n-2} = \frac{RSS/1}{ESS/(n-2)}$$

Equation A.22

Where RSS is the residual sum of squares, and ESS is the estimated sum of squares.

**Appendix IV.5.2 Regression Analysis Diagnostic Statistics**

These summary statistics are used to ascertain whether some of the assumptions about the residuals mentioned above, hold.

**II.5.2.1 Ramsey RESET**

This is a general test of specification error, devised by Ramsey in 1969 [Ramsey, 1969]. In a simple form, assume a regression equation as shown in equation A.14.

The test assumes that there will be no extra explanatory power if $\hat{Y}_i$ is added to the equation, given that the equation is correctly specified. If it is not, then adding another variable will increase the $R^2$ value, without adding any further explanatory power.

Therefore, to run a Ramsey RESET test, the regression must be run, initially to produce $\hat{Y}_i$. Then $\hat{Y}_i$, in some form, is introduced into the regression equation as an additional regressor. For example, the following regression could be run:

$$Y_i = \beta_0 + \beta_1X_i + \beta_2\hat{Y}_i^2 + \beta_3\hat{Y}_i^3 + \epsilon_i$$

Equation A.23

An F test is then run using the following test statistic:

$$F = \frac{(R^2_{new} - R^2_{old})/\text{new\_regressors}}{(1-R^2_{new})/(n-p)}$$

Equation A.24

where $p$ is the number of parameters in the new model.
Appendix IV.6 Breusch-Godfrey Test of Higher-Order Autocorrelation [Breusch, 1979, Godfrey, 1978]

If the disturbance term, $u_i$, is generated from a $p$th-order autoregressive scheme, such that:

$$u_i = \rho_1 u_{i-1} + \rho_2 u_{i-2} + \ldots + \rho_p u_{i-p} + \varepsilon_i$$  \hspace{1cm} \text{Equation A.25}

where $\varepsilon_i$ is a purely random disturbance term that conforms to all the assumptions of the disturbance term. The null hypothesis is that $H_0 : \rho_1 = \rho_2 = \ldots = \rho_p = 0$.

Breusch and Godfrey have shown that, on random sampling, the asymptotic distribution is:

$$(n-p) \cdot R^2 \sim \chi^2_p$$  \hspace{1cm} \text{Equation A.26}

II.6.1.1 Breusch-Pagan Test for Heteroscedasticity

From Breusch and Pagan [1979]: Consider a $k$ variable linear regression, model such that

$$Y_i = \beta_1 + \beta_2 X_{i2} + \ldots + \beta_k X_{ik} + u_i$$  \hspace{1cm} \text{Equation A.27}

Assume that the error covariance, \( \sigma_i^2 \), is described as

$$\sigma_i^2 = f(\alpha_i + \alpha_2 Z_{i2} + \ldots + \alpha_m Z_{im})$$  \hspace{1cm} \text{Equation A.28}

That is, \( \sigma_i^2 \) is some function of the nonstochastic variables $Z$s; some or all of the $X$'s can serve as $Z$s. Specifically, assume that

$$\sigma_i^2 = \alpha_1 + \alpha_2 Z_{i2} + \ldots + \alpha_m Z_{im}$$  \hspace{1cm} \text{Equation A.29}

That is, \( \sigma_i^2 \) is a linear function of the $Z$s. If $\alpha_2 = \alpha_3 = \ldots = \alpha_m = 0$, \( \sigma_i^2 = \alpha_1 \), which is a constant. Therefore, to test whether \( \sigma_i^2 \) is homoscedastic, test the hypothesis such that

$$H_0 : \alpha_2 = \alpha_3 = \ldots = \alpha_m = 0$$  \hspace{1cm} \text{Equation A.30}

However, the methodology above is based upon population parameters. These are not available when running the regression. Therefore, once the residuals have been obtained, the maximum likelihood estimator of \( \sigma^2 \), \( \hat{\sigma}^2 \) is calculated, such that
The variable $\pi_i$ is constructed, defined as
\[ p_i = \frac{\hat{\pi}_i^2}{\hat{\sigma}^2} \]  
Equation A.32

The calculated variable $\pi$ is then regressed onto the $Z_s$ as
\[ p_i = \alpha_1 + \alpha_2 Z_{2i} + \ldots + \alpha_m Z_{mi} + \nu_i \]  
Equation A.33

where $\nu_i$ is the residual term for this regression. The explained sum of squares of the residuals from Equation A.33 is then calculated, and then define
\[ \Theta = \frac{1}{2} (ESS) \]  
Equation A.34

Assuming $u_i$ are normally distributed, it is possible to show that if there is homoscedasticity and if the sample size $n$ increases indefinitely, then
\[ \Theta \sim \chi^2_{m-1} \]  
Equation A.35

II.6.1.2 Jarque-Bera Test for Normality

From [Gujarati, 1992]

The Jarque-Bera test (JB) for normality is an asymptotic, or large sample, test based on the OLS residuals. The test computes the skewness and kurtosis of the distribution, measures the OLS residuals and uses the following test statistic:
\[ JB = n \left[ \frac{S^2}{6} + \frac{(K-3)^2}{24} \right] \]  
Equation A.36

where $S$ represents skewness and $K$, kurtosis. Since the value of $S$ in a normal distribution will be zero, and the value of $K$ is three, any positive value of $(K-3)$ represents excess kurtosis. Under the null hypothesis that the residuals are normally distributed, Jarque and Bera showed that, asymptotically, the JB statistic follows the $\chi^2$ distribution with two degrees of freedom.
This appendix has presented an overview of some of the most common complexity metrics and their derivation. It has also covered simple linear regression, as it was applied in the pilot stage of the research.
References:


Halstead, M. H. [1977], Elements of Software Science, Elsevier North-Holland, New York


Appendix IV

Oviedo, E. [1980], “Control Flow, Data Flow and Program Complexity”, *Proceedings COMPSAC 80*, 146-52


Yourdon, E., Constantine, L.L. [1979], *Structured Design*, Prentice Hall, Englewood Cliffs, NJ
Appendix V
Appendix V

Example Questionnaire

For confidentiality reasons, this example questionnaire is not exactly the same as the one that was sent out. The layout has been slightly changed, and the questionnaire was printed on headed paper with security classification displayed.

This is similar to the first questionnaire that was sent out. The second questionnaire was larger. However, it was extracting data about current and future business requirements and capabilities. Removing company confidential material from the questions renders the questionnaire meaningless, and so it is not included here.
Appendix V

User Questionnaire

This questionnaire is solely for research purposes. It is not for use to 'check up' on your activities. It is intended to use information gained to improve the service. The questionnaire can be confidential, and you do not have to fill in your name, department, or job title. However, it would be very useful for research purposes if you did.

Once the questionnaire is completed, could you send it to Mark de Chazal either via email, or via internal mail [internal mail address]

<table>
<thead>
<tr>
<th>Name:</th>
<th>Department:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job Title</td>
<td></td>
</tr>
</tbody>
</table>

**Question 1:**

Which of the following applications do you currently use? [Please tick box]

<table>
<thead>
<tr>
<th>System 1</th>
<th>System 2</th>
<th>System 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System 4</th>
<th>System 5</th>
<th>System 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
</tbody>
</table>

**Question 2:**

Do you use these applications daily, weekly, or monthly? [Please enter system names in space provided]

<table>
<thead>
<tr>
<th>Daily</th>
<th>Weekly</th>
<th>Monthly</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Question 3:**

When did you last use System 1?: [Please tick box]

<table>
<thead>
<tr>
<th>Today</th>
<th>Yesterday</th>
</tr>
</thead>
<tbody>
<tr>
<td>□</td>
<td>□</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Last week</th>
<th>Last month</th>
</tr>
</thead>
<tbody>
<tr>
<td>□</td>
<td>□</td>
</tr>
</tbody>
</table>
Question 4:
When you used system 1, roughly for how long did you use it? [Please tick box]

<table>
<thead>
<tr>
<th>Time</th>
<th>0</th>
<th>15 minutes</th>
<th>0</th>
<th>1 hour</th>
<th>0</th>
<th>&gt;3 hours</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;10 minutes</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30 minutes</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 ½ hours</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 hours</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Don’t remember</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Question 5:
Approximately how long do you estimate you spend using System 1 per year? [Please tick box]

<table>
<thead>
<tr>
<th>Time</th>
<th>0</th>
<th>50 hours</th>
<th>0</th>
<th>100 hours</th>
<th>0</th>
<th>300 hours</th>
<th>0</th>
<th>&gt;400 hours</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;10 hours</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25 hours</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>75 hours</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200 hours</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>400 hours</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Question 6:
How do you normally access the applications you use? [Please tick box]

<table>
<thead>
<tr>
<th>Access</th>
<th>0</th>
<th>UNIX</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Question 7:
Lastly, do you have any comments about any of the systems you use?

Thank you very much for completing this questionnaire.
If you have any further questions, or would like to see how this research is being used, please contact:
Mark de Chazal
Internal phone number: XXXXXX
Email: XXXXXXXXXX
Internal Mail Code: XXXXXXX
Location: Xxxxxxxx

306
Appendix VI

Record of Contribution

Head of Support
Dear Mr Dawson

Record of Contribution of the Sponsored PhD Student, Mark de Chazal

Towards the end of 1998 the Submarines Business was being directed towards a range of corporate data management systems. Initial investigations raised concerns over the suitability and capability of these systems to accept our legacy data and provide a platform for the future.

In view of this we decided to sponsor a PhD student to address the strategic decision making process for legacy systems and thereby ensure that Rolls-Royce extracts the maximum benefit from them as part of a cohesive cost effective strategy.

Since the start of the sponsorship, Mark has performed admirably in this role and has been involved in many projects of strategic value to the Rolls-Royce Submarines Business. These have included:

- The production of legacy system management options
- Capability definition groups
- The production of several capability acquisition strategies
The output of the various focus groups has enabled the Submarines Business to provide an objective presentation regarding option recommendations. These recommendations have often followed a route that does not align with corporate policy, but it has been a defensible position, with clear capability, cost, and timescale requirements and benefits.

Rolls-Royce have financially and technically supported the PhD, and have found the resultant work to be of great benefit to Rolls-Royce, underpinning strategic recommendations within the Submarines business.

Yours sincerely

WJ Simmons
Head of Support Capability
Rolls-Royce
Appendix VII

Publication List

Publications in the Public Domain:


