Managing Application Software Suppliers in Information System Development Projects

by

Angus Gonghua Yu

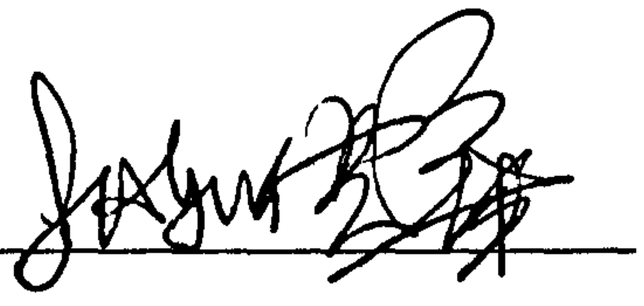
Thesis submitted for the degree of
Doctor of Philosophy

Department of Management and Organisation
University of Stirling
Stirling FK9 4LA, Scotland

November 2003

DECLARATION

This thesis has been composed in its entirety by the candidate. Except where specifically acknowledged, the work described in this thesis has been conducted independently and has not been submitted for any other degree.

Signature of the candidate: 

Date: 21/5/2004

DEDICATION

To my parents, my wife and my children.

ACKNOWLEDGEMENT

I would like to express my gratitude to all who have helped me in my career to date that has, in a sense, culminated in this thesis. In particular, I would like to thank my former colleagues and employers who supported me during my years of work and study. I would like to thank my supervisor John A. Bowers who may have spent more than the expected time for supervising a part-time PhD student. I benefited from many discussions we have had. In addition, I am grateful to some of the other colleagues in the University of Stirling who have proven to be valuable sources of encouragement during the course of my research.

I am indebted to my family and friends for supporting me in many ways throughout my years of study. In particular, they have tolerated me for not fulfilling my duties as a family member and as a friend in many an occasion. I hope that I can make it up in the future.

ABSTRACT

Information system development (ISD) projects have been associated with the "software crisis" for over three decades. A set of common "root causes" has often been cited in literature with corresponding "solutions". Yet the overall project success rate has remained low, resulting in a paradox of many solutions and little progress over the years. This study examines the management of application software acquisition from external suppliers in ISD projects. Three case studies are documented based on participant observation with complete membership roles. After within-case analyses highlighting issues in individual cases, cross-case analyses are conducted, first to identify a pattern of ISD project challenges and then to search for their explanations. Concepts from agency theory, contract theory and product development literature are used in the process of diagnosing root causes behind the observations. The proposed explanation is that the Traditional Systems Development Framework (TSDF), characterized by competitive-bidding-monopolized-development, underlies the identified root causes. Accordingly, competitive development is suggested as an alternative approach. Following the "Inference to the Best Explanation" (IBE) analytical strategy, the suggested approach is subject to two contrastive analyses, first with the pre-packaged software development and then with the construction industry, to demonstrate that the suggestion is a "warranted inference". Further analogical analyses illustrate the feasibility of development competition for software product development. A Performance-Based Systems Development Framework (PBSDF) is outlined as a tentative implementation of the suggested competitive development approach for ISD projects supported by risk-sharing supplier contract and a relative product evaluation approach. A number of future research implications are described as a result of this study after summarizing the research contributions. (263 words)

TABLE OF CONTENTS

Chapter 1: Information System Development Projects Involving Application Software Suppliers 1

1.1 Introduction 1

1.2 IT Products and Services Market 1

1.3 Information Systems and ISD Projects 4

1.4 ISD Project Challenges and Research Motivation 6

1.5 Research Objectives 8

1.6 The Scope of the Research and The Thesis Outline 9

1.7 Summary 12

Chapter 2: Research Methodology 13

2.1 Introduction 13

2.2 Philosophical Schools and Research Approaches 14

2.3 Research Strategy and Research Process 18

2.3.1 Research Strategy 19

2.3.2 Research Process 21

2.4 Research Design 28

2.4.1 The Adopted Research Strategy 28

2.4.2 Research Components 30

2.5 Data Collection Based on Participant Observation 33

2.5.1 Participant Observation as a Research Method 34

2.5.2 Participation in the Case Study Projects 34

2.5.3 The Structure of Case Study Composition 36

2.6 From Observation to Understanding 37

2.6.1 Inference to the Best Explanation (IBE) 38

2.6.2 Criteria of "The Best" Explanation 40

2.6.3 The IBE Process and Its Application in This Study 41

2.7 Research Ethics 43

2.8 Summary 46

Chapter 3: The "Software Crisis" and the Current Solutions 47

3.1 Introduction 47

3.2 Defining an Information System 48

3.2.1 Current IS Definitions 48

3.2.2 Hardware 49

3.2.3 System Software and Application Software 49

3.2.4 Data as Part of an Information System 50

3.2.5 Documentation as Part of an Information System 52

3.2.6 Business Processes as Part of an Information System? 52

3.2.7 Users as Part of an Information System? 54

3.2.8 A Recommended Definition of an Information System 54

3.3 Success and Failure Criteria of ISD Projects 55
3.3.1 The Traditional Approach on Project Success Criteria 56
3.3.2 Project and Product 58
3.3.3 A Proposed Approach to Project Success/Failure Criteria 60

3.4 The "Software Crisis": Published Cases 62
3.4.1 The Computer Aided Despatch (CAD) System Project 63
3.4.2 The CONFIRM Project 64
3.4.3 The CAPSA Project 67

3.5 The "Software Crisis": Surveys and Reviews 70

3.6 Information System Development Methodologies 75
3.6.1 What is an ISDM? 75
3.6.2 The Many ISDMs 76

3.7 The Paradox of Many Solutions and Little Progress 78
3.7.1 "Why they do not practise what we preach?" (Humphrey, 1998) 80
3.7.2 Have we got the "message" right? 82
3.7.3 Why might we have got the message wrong? 85

3.8 Summary 86

Chapter 4: Case 1 – The WebShop Project (April 2000 – March 2001) 88

4.1 Background 88

4.2 Project Organisation and Resources 89

4.3 Selecting and Contracting Application Software Suppliers 91
4.3.1 Selecting and Contracting NQ 91
4.3.2 Selecting and Contracting MSS 93
4.3.3 Work Allocation Between NQ and MSS 94
4.3.4 An Additional Supplier IFG 94

4.4 Suppliers' Deliverables and Milestones 95
4.4.1 Period 1: From April to mid September 2000 (SLC, 9, 10, 11) 96
4.4.2 Period 2: From Mid September to December 2000 97
4.4.3 Period 3: From Jan 2001 to March 2001 99

4.5 Project Budgeting and Cost 100

4.6 Software Product Quality Control 102
4.6.1 Requirements Management 103
4.6.2 Product Acceptance 104
4.6.3 Quality of NQ Deliverables 106
4.6.4 Quality of MSS/IFG Deliverables 107

4.7 Project Issues and Within-Case Analysis 108
4.7.1 Project Organisation 108
4.7.2 Supplier Selection 109
4.7.3 Partnership Approach 110
4.7.4 The Timeboxing Method 111
4.7.5 Requirements Management and Variation Control 112
4.7.6 Resource Capacity Planning 113
4.7.7 Risk Management 114
4.7.8 Project Success or Failure? 115

4.8 The WebShop Project Chronology 115

4.9 Internal References 118

Chapter 5: Case 2 – The Gas Meter Reading Management System (March 2001 - October 2001) 124

5.1 Background 124

5.2 SubB's Requirements 125

5.3 Options Available and Supplier Selection 127

5.4 Project Organisation and Resources 129

5.5 Contracting with AES 130
5.5.1 Deliverables and Delivery Time 131
5.5.2 Contract Price and Payment Schedule 131
5.5.3 Acceptance Criteria 132
5.5.4 Contract Administration 133

5.6 Project Budget and Cost Control 134

5.7 Product Quality Control and Acceptance Testing 135

5.8 Actual Project Progresses 136

5.9 Project Issues and Within Case Analysis 137
5.9.1 Supplier Selection 137
5.9.2 System Architecture 137
5.9.3 Data Integration 139
5.9.4 Product Acceptance Testing 140
5.9.5 Software Product Documentation 141
5.9.6 Risk Management 142
5.9.7 Post-Implementation Review 143
5.9.8 Project Success or Failure? 143

5.10 MPS Project Chronology with Events Relating to GMRM 144

5.11 Internal References 146

Chapter 6: Case 3 – Building a Low Voltage Connectivity Model (July 2001 to March 2002) 150

6.1 Background 150

6.2 SubC Requirements Regarding LVC 152

6.3 Project Organisation and Resources 153

6.4 Supplier Selection and Contracting 154
6.4.1 Selecting the Application Software Supplier 154
6.4.2 Contracting with SC1 157

6.5 Project Budget and Cost Control 157

6.6 Product Quality Control and Product Acceptance 159

6.7 Actual Project Progresses 160

6.8 Key Issues and Within-Case Analyses 161
6.8.1 Poor System Performance 161
6.8.2 Poor Interoperability between OMSTool and Data 163
6.8.3 Project Documentation 163
6.8.4 SC1's Poor Response to Problems 164

6.8.5 Risk Management 165
6.8.6 Project Success or Failure? 167
6.9 Project Chronology 167
6.10 Internal References 169
Chapter 7: Cross-Case Analysis Part 1: Describing the Pattern of ISD Project Challenges 173
7.1 Introduction 173
7.2 Overview of the Three Participant Case Studies 173
7.3 Managing Supplier Selection and Contracting 175
7.3.1 Supplier Selection 175
7.3.2 Time Gap Between Supplier Selection and Contracting 177
7.3.3 Supplier Contract Management 178
7.3.4 Client-Supplier Relationship 180
7.4 Managing Software Product Development 181
7.4.1 Requirements Management 181
7.4.2 Requirements-Driven vs. Delivery-Led Development 182
7.4.3 Supplier Capacity Management 184
7.4.4 Interoperability of Application Software and Data 186
7.4.5 Software Documentation and Its Quality 187
7.5 Evaluating Supplier Product 188
7.6 Project Risk Management 190
7.7 The Pattern of ISD Project Challenges 191
7.8 Summary 193
Chapter 8: Cross-Case Analysis Part 2: Explaining the Pattern of ISD Project Challenges 195
8.1 Introduction 195
8.2 Client and Supplier Relationship 196
8.2.1 Client-Supplier as Partners? 196
8.2.2 Principal-Agent Relationship 199
8.3 Misconception About Requirements Management 204
8.3.1 The Current Explanation for Incomplete Requirements 204
8.3.2 An Alternative Explanation: The Kano Model 206
8.4 Switching Cost and Its Impact on Supplier Behaviour 208
8.5 Incomplete Contracts 211
8.5.1 The Concept of Incomplete Contracts 212
8.5.2 Contracts Used in ISD Projects 215
8.6 Inseparability of Design and Development 217
8.7 Risk and Uncertainty 219
8.7.1 The Concepts of Risk and Uncertainty 219
8.7.2 Software Development Uncertainty and Its Consequences 221
8.7.3 The Management of Uncertainty in Projects 223
8.8 Product Evaluation 226

8.8.1 Two Approaches of Product Evaluation 226
8.8.2 Two Stages of Product Evaluation in ISD Projects 228

8.9 The Traditional Systems Development Framework 229
8.9.1 Describing TSDF 230
8.9.2 Modelling TSDF with Game Theory 238

8.10 A Critique of Iterative Development Approach 242
8.10.1 The Iterative Approach and Its Contributions 242
8.10.2 The Converging Argument for the Iterative Approach 243
8.10.3 Inadequate Considerations for Root Causes 245

8.11 A Unified Explanation 247

8.12 Summary 249

Chapter 9: Comparing ISD Projects with Other Types of Product Development 251

9.1 Introduction 251

9.2 Contrasting PPSW Market with ISD Projects 251
9.2.1 PPSW – Software Development Can Be Successful 252
9.2.2 Different Market Mechanisms and Different Development Approaches 256

9.3 Construction Projects: An Appropriate Metaphor? 259
9.3.1 Contrasting Construction and ISD projects 259
9.3.2 A Critique of the Construction Metaphor 264

9.4 Comparing ISD with Manufacturing Product Development 266
9.4.1 Design versus Manufacturing 267
9.4.2 "Black Box" Component Development in Manufacturing 268
9.4.3 Lessons from the SLMP Design Process 269

9.5 Open Source Software Development 271
9.5.1 OSS Development Process 272
9.5.2 Capitalising on Software Virtuality 274

9.6 Summary 275

Chapter 10: Performance-Based Systems Development Framework 277

10.1 Introduction 277

10.2 Pre-Contract Understandings 278
10.2.1 Pre-Contract Understanding: The Concept of An Information System 278
10.2.2 Pre-Contract Understanding: Incomplete and Changing Requirements 279
10.2.3 Pre-Contract Understanding: Price Changes 279
10.2.4 Pre-Contract Understanding: Delivery Time Adjustments 280

10.3 An Outline of PBSDF 281
10.3.1 PBSDF - Part I: A More Complete Contract 281
10.3.2 PBSDF – Part II: Development Competition 285
10.3.3 PBSDF – Part III: Product Evaluation 287

10.4 The Theoretical Basis of λ 288

10.5 Empirical Estimates of λ 290
10.5.1 Estimating λ with Data from Annual Accounts 290
10.5.2 Estimating λ by Labour Cost Data from SLC 291

10.6 Practical Considerations for Setting λ 293
10.6.1 Varying λ According to the Level of Competition 293
10.6.2 Varying λ According to Project Length 294
10.6.3 Varying λ According to Project Risk 294
10.6.4 Varying λ According to Supplier's Product Maturity 294

10.7 Implementing PBSDF with Two Suppliers 295
10.7.1 ISD Project Organisation for PBSDF-TSS 296
10.7.2 Stage 1: Client Preparing "Request for Proposal (RFP)" 297
10.7.3 Milestone 1: RFP Issued 298
10.7.4 Stage 2: Pre-Contract Proposals 298
10.7.5 Milestone 2: Client Committing to Contract with Two Suppliers 298
10.7.6 Stage 3: Software Product Development 299
10.7.7 Milestone 3: Suppliers Release Software for Evaluation 300
10.7.8 Stage 4: Client Conducts Product Evaluation and Considers Options 300
10.7.9 Milestone 4: Client Announces Product Evaluation Result 301
10.7.10 Stage 5: System Implementation and Post Implementation Activities 301

10.8 Cost-Benefit Analysis of PBSDF-TSS 301
10.8.1 Total Cost with TSDF 301
10.8.2 Total Cost with PBSDF-TSS 303
10.8.3 Comparing C_0 and C_1 304

10.9 Game Theoretic Modelling of PBSDF-TSS 307
10.9.1 Subgame for Contract Bidding 307
10.9.2 Subgame for Development Competition: The Effect of Switching Cost ... 309
10.9.3 The Contract Between a Client and a Supplier 310
10.9.4 The Effect of Development Competition On An Individual Supplier 312

10.10 Alternative Implementations of PBSDF 313
10.10.1 Implementing PBSDF with a Single Supplier 314
10.10.2 Implementing PBSDF with Multiple Suppliers 315

10.11 Discussions on PBSDF Implementation 316
10.11.1 Prototyping 317
10.11.2 Requirements Management 318
10.11.3 Quantified Product Evaluation 318
10.11.4 PBSDF for Large ISD Projects 319
10.11.5 The Diversity of Software Systems 319

10.12 Summary 320

Chapter 11: Conclusions and Implications for Future Research 322

11.1 Overview of the Study 322

11.2 Limitations of The Research 323
11.2.1 Limitations with Data Collection 323
11.2.2 Limitations with Analysis 323

11.3 Main Contributions 324
11.3.1 Confirming Widespread Challenges in ISD Projects 324
11.3.2 Root Causes of ISD Project Challenges 324
11.3.3 PBSDF - A Proposed Solution 327
11.3.4 Applicability of PBSDF to Different Types of ISD Projects 329

11.4 Other Contributions of this Research 330
11.4.1 Project Success and Failure Criteria 330

11.4.2 Contribution to Project Management Research 330
11.4.3 Project Risk Management 331

11.5 Main Areas for Future Studies 332
11.5.1 Disseminating the Research Findings 332
11.5.2 Action Research Based Case Studies 333
11.5.3 Adopting PBSDF for Internal Software Development 334

11.6 Other Areas for Future Studies 334
11.6.1 Project Success Criteria 335
11.6.2 Less Distorted Bidding Prices in PBSDF 335
11.6.3 Realistic Project Feasibility Assessment 336
11.6.4 Modelling ISD Projects with Systems Dynamics 337
11.6.5 The Interaction of Process and Data 338
11.6.6 Value Management and Value Engineering 339

11.7 Summary 339

References 341

Appendix A: Acronyms Used in The Thesis 368

Appendix B: 40 Root Causes Compiled by (Smith, 2001, pp. 18-19) 372

Appendix C: Quantitative Risk Analysis Matrix (QRAM) Template 374

LIST OF FIGURES

Figure 1.1: An ISD project organisation with hardware and software suppliers 6
Figure 2.1: Research process spiral 22
Figure 3.1: The lifecycle of data in an information system 51
Figure 3.2: A schematic IS architecture 55
Figure 4.1: The WebShop project organisation 90
Figure 4.2: The initially intended architecture for the WebShop System 96
Figure 4.3: Architectural sketch of the delivered WebShop system by February, 2001 98
Figure 5.1: Systems related to gas operations in SubB before the MPS Project 126
Figure 5.2: Simplified architecture for the proposed solution 129
Figure 5.3: The MPS project organisation 130
Figure 6.1: Indicative organisational chart for The IIP/LVC Project 153
Figure 8.1: The Kano Model 206
Figure 8.2: The game of software development in TSDF 240
Figure 8.3: The converging argument for the iterative development 243
Figure 8.4: Fixing time and resources in "timeboxes" and varying functionality 245
Figure 9.1: Quality-adjusted price indexes for PPSW, CBSW and internal software development, with the 1996 price index set as 1.0 253
Figure 9.2: Comparing PPSW, COTS and CBSW in respect of pre-Contract and post-contract development and other activities before providing client with benefits. 255
Figure 10.1: Five possible outcome scenarios of a software development contract 282
Figure 10.2: A typical ISD project organisation for implementing PBSDF-TSS 296
Figure 10.3: Direct payment difference for software product (Δ_s) between using PBSDF-TSS and TSDF 305
Figure 10.4: The game of client-supplier contract in PBSDF-TSS 310

LIST OF TABLES

Table 3.1: Project success/failure categories for projects aiming to develop products 61
Table 3.2: Software projects success rates by company sizes 70
Table 3.3: A survey of surveys on IT Project success rates 71
Table 3.4: Classifications of ISDMs based on their approaches 77
Table 4.1: WebShop key milestones as of early October 2000 97
Table 4.2: Shifting of the two key milestones of WebShop Project from Jan 2001 to Feb 2001 100
Table 4.3: Estimated (Costs) at Completion (EAC) for three tasks from June 2000 to March 2001 101
Table 4.4: Suggested acceptance criteria 105
Table 4.5: Defects/issues statistics between 14/11/2000 and 24/11/2000 107
Table 5.1: Indicative cost summary of the two supplier options 128
Table 5.2: AES' deliverables schedule dated 28/03/2001 131
Table 5.3: AES' initial payment schedule 132
Table 5.4: MPS Project cost estimation and monitoring 134
Table 5.5: GMRM delivery progress as compiled on 26/10/2001 136
Table 6.1: Supplier SC1's perceived compliance summary 155
Table 6.2: Supplier SC1A's perceived compliance summary 155
Table 6.3: Comparing the bids during supplier selection 156
Table 6.4: Milestone payment schedule between SubC and SC1 157
Table 6.5: LVC Subproject budget breakdown 158
Table 6.6: Performance testing result: single user vs. multiple users on 17/01/2001 162
Table 6.7: Risk Assessment on the Software Availability from July 2001 to May 2002 . 166
Table 7.1: An overview of the three cases recorded in Chapters 4, 5 and 6 174
Table 7.2: Comparison of the risks related to the required software products and the suggested mitigating actions in across three projects 191
Table 8.1: Dynamic client-supplier contract changes in the Libra Project 224
Table 8.2: Two stages of product evaluation in ISD projects 229
Table 8.3: Types of games 239
Table 10.1: Service costs as percentages of the corresponding service revenues 290
Table 10.2: SLC's direct labour costs as percentage of PCR (with 23% off the market rates) 291
Table 10.3: Direct labour costs as percentages of the average market full charge rates 292
Table 10.4: Modelling the effect of private switching costs in PBSDF-TSS 310

Chapter 1: Information System Development Projects Involving Application Software Suppliers

1.1 Introduction

This section introduces Chapter 1 that defines the research subject for the whole thesis: the management of application software suppliers in information system development (ISD) projects. To provide a context for the research subject, the market of information technology (IT) products and services is surveyed (Section 1.2). Section 1.3 describes briefly the organisational need for information systems and the necessity of ISD projects, often involving external application software suppliers. Section 1.4 highlights the research motivation based on reported ISD project challenges, and more specifically the author's personal experience of troubled IT projects. Section 1.5 outlines the research objectives while Section 1.6 details the scope of the research and the thesis outline.

1.2 IT Products and Services Market

The global IT spending excluding telecommunications was expected to be $1.21trillion in 2001 (Kumar, 2001; Wright, 2001). This figure was expected to sustain 9.8 per cent compound annual growth rate between 2001 and 2005 to $1.746 trillion in 2005. The estimates may vary from one source to another. The message is the same: the expenditure on IT products and services is and will be substantial.

The IT expenditure reported above is defined as "money spent on purchasing hardware, software, or services of a commercial grade" (Wright, 2001). Accordingly, the IT market may be divided into the following segments:

- Hardware products and services;

- Software products and services.

The sub-market of software products and services may be further divided into the following segments:

- Pre-packaged software (PPSW) products;
- Commercial Off-The Shelf (COTS) software products;
- Custom-built software (CBSW) products;
- Software integration and other system development services;
- Software and system support and maintenance services, including enhancement development.

Various types of software products are developed and integrated with hardware products to become information systems with the help of software integration and other system development services. The support and maintenance services are required to keep information systems operational.

The concepts of PPSW, COTS and CBSW products may be further clarified. PPSW refers to packaged software applications ready to be used. Typically they are sold through distribution channels without software development suppliers being directly involved in their installation and integration. PPSW are designed and developed to interoperate with users' data without having to be modified. Examples include products like operating systems (Windows 95, Windows 98, Windows 2000, Windows XP, Unix and Linux) and general-purpose software applications like Microsoft Word and Microsoft Excel. CBSW products are designed and developed for clients with designated purposes. They may interoperate only with specific clients' data. Therefore CBSW products are usually not distributed outside of the target client organisations. An example of CBSW is SABRE, a computerised reservation system developed and deployed by AMR (Hopper, 1990). COTS products are somewhere in between PPSW and CBSW. COTS products are developed to serve a group of clients, typically defined by industry sectors or business requirements. COTS products may have to be

modified to work with clients' data and business processes. Alternatively, the clients' data and business processes may have to be converted to operate with COTS. Enterprise resource planning (ERP) software packages offered by SAP, Oracle and PeopleSoft (Ferris, 1999) are good examples of COTS products. The three product types can be envisaged on a continuous spectrum, with PPSW at one end and CBSW at the other while COTS somewhere in between depending on the level of development in each COTS product (see e.g. Brownsword et al., 1998; D'Adderio, 2000; see also Section 9.2).

There are other categories that are derivatives from the above in one way or another, for example, Military Off-The-Shelf (MOTS) and Government Off-The-Shelf (GOTS) software products (Kohl and Oberndorf, 2000). Since MOTS and GOTS would have been either bought or custom-built in the first place, they are not listed as separate categories above. Another often-encountered software type is the Open Source Software (OSS). OSS is different in its method of development and distribution (OSI, 2001, 2002; Lerner and Tirole, 2000). According to the level of customisation required for use in information systems, OSS may be regarded as either PPSW or COTS.

The success of information technology as a whole has been significant since the 1950s. This is most evident in the hardware advancement. The famous Moore's Law, first stated by Gordon E. Moore in 1965, forecasts that the number of transistors per integrated circuit would double every 18 months. The Moore's Law still holds true to this day (Stewart, 2002). While the hardware performance has increased by leaps and bounds, their prices have decreased dramatically to be more affordable (Levy, 1987; Brooks, 1995; Dooijes, 1999).

The software products and services sector has also seen a significant amount of progress. PPSW provides a vast array of products when it comes to the system software and generic utilities like operating systems, compilers and editors etc (Brooks, 1995). There are enduring

successes like Unix as an operating system. Fortran and COBOL initially designed in the late 1950s and early 1960s are still in use as programming languages. Though there are concerns that there has been limited progress on system software research (Pike, 2000), there are commercial success stories like Microsoft Windows series of Graphic User Interface (GUI) based operating systems. The available PPSW products may not be perfect (Kaner and Pels, 1998), and may not have exploited the available hardware capability fully (Dooijes, 1999), but they certainly have made it possible for the mass population to enjoy the wonders of the information society.

1.3 Information Systems and ISD Projects

Software applications alone usually do not meet specific organisational needs. Instead they have to be integrated with other system elements to become information systems. A system can be defined as "an integrated set of elements that accomplish a defined objective" (INCOSE, 2000, p. 9). For an information system, the elements might include hardware, system software, necessary networking facilities, application software and data etc (for a more detailed discussion of information system elements, see Section 3.2). An information system does not have to be based on computer technology. However, within the context of this thesis, the term "information system" will be used to refer to computer-based ones.

Information systems have been used in organisations to serve a wide range of purposes. For example, they might be used:

- To automate otherwise manual and repetitive business processes. Data processing systems developed in the 1960s and 1970s are among this category (Friedman, 1989);
- To assist management decision-making (i.e. decision support systems);
- To gain competitive advantage for companies in the market place, e.g. SABRE (Hopper, 1990);
- To assist communication with external organisations (e.g. supply chain systems).

Organisations have increasingly found themselves operating in an environment that has been characterized as a global and digital economy (Avison and Fitzgerald, 2003), which is ever more dynamic and competitive. Information systems are potentially powerful tools that can help organisations to survive, and indeed, thrive on changes. To meet such demands, information systems have to be developed cost-effectively and with high quality (including high reliability, easy maintainability etc).

Organisations develop new information systems by acquiring or developing the necessary system elements. Since information system development is usually a one-off undertaking involving a set of interrelated activities, they are often managed as projects. PMI (2000) defines a project as "a temporary endeavor undertaken to create a unique product or service". This definition matches well what an ISD project intends to achieve, namely, creating a unique information system to serve the needs of an organisation. ISD projects should be differentiated from other information systems related projects where existing systems are modified (e.g. upgrading an existing system platform to a newer version). ISD projects are also referred to as software projects or IT projects in the press and daily conversations. When there is no confusion of the meanings, these terms will be occasionally used in this thesis, especially when quoting from other sources.

An ISD project involves activities of acquiring and developing necessary system elements and integrating them into coherent information systems for specific organisational needs. The defining system element in an information system is usually the core application software, also referred to as the software application. An organisation may develop the application software entirely with internal resources, as AMR did in developing SABRE over decades (Hopper, 1990). More often, however, external suppliers are required to supply COTS/CBSW software. A typical ISD project might be organised as Figure 1.1 shows with various

suppliers. In Figure 1.1, the project is led by a project manager, who in turn reports to a client project sponsor, usually a senior manager in the client organisation. The ISD project team is responsible for managing requirements with the client organisation on the one hand and working with suppliers to acquire necessary system elements, including hardware, system software and application software.

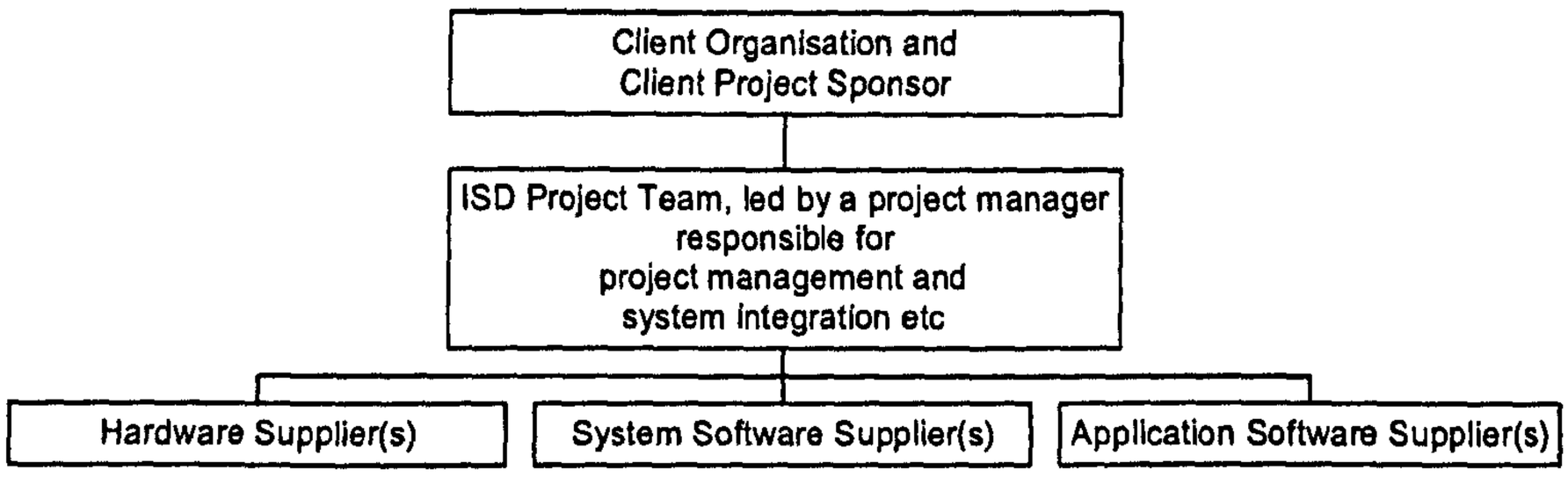


Figure 1.1: An ISD project organisation with hardware and software suppliers

When a group of projects are required to achieve more substantial organisational objectives, the projects may be managed together as a programme. An example is the NHS National IT Programme that is planned to run from 2003 for five years (Arnott, 2003a). This thesis focuses on the activities at the project level between a project team and the application software supplier(s). There may be one or more layers of suppliers from which an application software supplier acquires software components and tools. Occasionally, issues (e.g. licence agreements) arising from these lower tiers of suppliers are relevant to client organisations. Such issues are often resolved through the application software suppliers and therefore are not dealt with separately in this study.

1.4 ISD Project Challenges and Research Motivation

ISD projects have posed significant challenges to both IT professionals and their client organisations in the last few decades. According to Standish Group (1995), on average only 16.2% of software projects are completed on time and within budget. At the time of that study, $250 billion was being spent annually in the United States alone on 175,000 projects.

An estimated $81 billion of these projects would eventually be cancelled. The report states that "software development projects were in chaos". There have been many allegations of the "software crisis" over the years (e.g. Naur and Randell, 1968; Brooks, 1975, 1995; Glaseman, 1982; Abdel-Hamid and Madnick, 1991; Gibbs, 1994; Flowers, 1996; Jones, 1996; Collins, 1998; Glass, 1998, 1999; Bronzite, 2000; Smith, 2001; Avison and Fitzgerald, 2003).

The motivation for the research, however, has not come directly from reading the historical accounts of project challenges. Rather it has originated from the author's personal experience of ISD projects. The author was in the procurement profession in the manufacturing sector for over seven years before joining an information systems division of a FTSE100 company (referred to as UKOne throughout the thesis) in May 1999. One year later, the division was spun-off as a separate company (referred to as SLC throughout the thesis). As part of the separation deal, SLC secured a five-year contract to provide support to the IT infrastructure and many of the existing systems within UKOne. In addition, the deal entitled SLC to be treated as a preferred supplier for system development and integration in UKOne's ISD projects.

The author was assigned to one of the ISD projects undertaken by SLC for UKOne in July 2000 as a test manager. The project was initiated in April 2000. Therefore, at the time of the author's joining, the initial preparation had already been undertaken. The project was to be completed by November 2000, but did not end until March 2001. Immediately afterwards, the author was assigned to a new project, which was going through a project proposal stage. The author first acted as an assistant to the proposal manager, and then as the project test manager. The project was planned to complete by August 2001 but did not close down until the end of October that year. Both of these projects relied on external suppliers to provide core application software products. Both projects experienced significant challenges, some of which seemed similar in nature. In particular, communications with suppliers were difficult,

involving heated exchanges in project meetings between professionals who were otherwise calm and mature. Also, software products from the suppliers were of poor quality, not satisfactory from the clients' viewpoint. Yet there seemed to be few options available to the clients other than accepting the products and paying the suppliers in full. Such client-supplier relations seemed entirely different from what the author had experienced previously in the manufacturing sector. The author was curious at the time regarding what had made the client-supplier relations in the software industry so different. The observed contrast of the two different client-supplier relation types is the basis of the "study proposition" discussed as part of the research scope (Section 1.6) and the research design (Section 2.4.2).

The author consulted with his colleagues at the time and met with a general sense of cynicism and resignation. A typical response was "it has always been like that!" A casual glance of ISD project related books like Brooks (1975, 1995) and Collins (1998) seemed to confirm the widespread cynicism but there had appeared to be no specific discussion on supplier management in ISD projects. The author decided to embark on a research journey. The author registered with the University of Stirling in September 2001 for a research degree on a part-time basis. In the meantime, the author continued with the full-time employment in SLC. The thesis is the result of the research effort. The two projects mentioned above, together with another one participated by the author from November 2001 to March 2002, have been documented partly based on the author's personal experience, and more significantly based on the available project documents. The three projects are presented as Chapters 4, 5 and 6. The research methodology is considered in Chapter 2.

1.5 Research Objectives

This study aims to develop a better understanding of ISD project challenges so that a new potential explanation might be found for the challenges and corrective actions can be formulated accordingly to guide project management. More specifically, the objectives are:

- To confirm some of the common symptoms of the ISD project challenges;
- To understand why some of the current solutions may have not been successful;
- To seek reasoned explanation(s) for the observed ISD project challenges;
- To develop a new potential solution based on the above explanation(s);
- To identify future research directions related to the proposed solution and other opportunities that may arise as a result of this study.

The research strategy and design is described in Section 2.4.

1.6 The Scope of the Research and The Thesis Outline

To achieve the above objectives, this study focuses on the management of suppliers for the purpose of acquiring or developing application software within the context of ISD projects. Such software may be of COTS or CBSW types. The study does not address the development or acquisition of PPSW, though the PPSW market segment is examined as a contrast to COTS and CBSW in a comparative study in Chapter 9. The research does not attempt to address the problems associated with outsourced IT support and maintenance, which is a subject of research on its own (see e.g. Lacity and Willcocks, 2000).

There are mainly three reasons for focusing on supplier management in ISD projects. First of all, a software application is at the heart of an information system. When the application is to be purchased, either as COTS or CBSW, from external suppliers, its success or failure directly affects the success or failure of the entire ISD project and the resultant information system (e.g. Page et al., 1993). Secondly, supplier management is related to many other aspects of project management. In PMBOK Guide (PMI, 2000), project management is broken down into the following nine areas:

- Project Integration Management
- Project Scope Management
- Project Time Management

- Project Cost Management
- Project Quality Management
- Project Human Resource Management
- Project Communications Management
- Project Risk Management
- Project Procurement Management

While the project procurement management includes important issues of supplier selection and contract management (PMI, 2000), it cannot be isolated from other aspects including the management of requirements (scope), time, cost, quality, communications and risk. The approach taken in this study aims to examine all aspects related to supplier management holistically. This is necessary due to the complexity and the uncertainty of software development process, which has led to difficulties in contract management. Lichtenstein (2002) reports "puzzles" in the study of software development contracting, which might be attributed to its approach of separating contracting from other complex issues like requirements management and product evaluation. The third reason for focusing on software supplier management is that the organisational boundary between a client and a supplier helps their communications to be formalised and commitments to be made specific and recorded. Such documentary evidence has been used extensively in the participant case studies in this thesis to establish project issues and forms the basis of the case studies.

The scope of the study excludes in-house ISD projects. The study assumes that the client has decided to "buy" COTS software for further development and integration, or to "buy" the management and technical expertise from suppliers for developing CBSW. The study does not intend to make any contribution to the debate about the pros and cons of "make" or "buy" approaches. However, the study is potentially relevant to internal ISD projects. This is because internal software development teams might be considered as part of an "internal value chain" in the same way as external suppliers form part of the "external value chain"

(Ward and Peppard, 2002). Any attempt to extrapolate the research findings to internal ISD projects, however, should be carried out with caution and should be rigorously validated by further research.

The study does not deal with the relationship between the client project sponsor and the ISD project team specifically. It can be argued that this set of relationship is also critical to the success of an ISD project. This can be the case especially when an ISD project is largely managed and staffed by external resources from outside the client organisation. However, for this study, it is assumed that the interests of the client and the ISD project team are aligned, either because of organisational allegiance or appropriate risk and reward contracts. It would be a separate study altogether to examine how such alignment of interests can be achieved.

The thesis is organised essentially into three parts. The first part consists of Chapters 1, 2 and 3. Chapter 1 provides the context of the research. Chapter 2 considers the research methodology. Chapter 3 surveys the literature relevant to ISD project challenges and some of the solutions available already. This part answers questions like why the research is necessary and how it will be carried out. The second part consists of Chapters 4 to 6, consisting three participant case studies. The third part of the thesis provides analyses of the root causes of the project challenges and proposes a management framework as a solution. Chapter 7 is Part 1 of the cross-case analyses, describing the pattern of ISD project challenges. Chapter 8 is Part 2 of the cross-case analyses, identifying the root causes for the pattern of ISD project challenges and proposing a unified explanation to the common ISD project challenges. The explanation is subjected to a number of comparative analyses in Chapter 9. Chapter 10 proposes a management framework as a potential solution based on the explanation. Chapter 11 concludes the thesis with an outlook for future research potentials.

1.7 Summary

This chapter defines the research subject for this thesis as the management of application software suppliers within the context of ISD projects. Due to operational, competitive and regulatory needs, organisations require information systems to improve efficiency and to gain or sustain competitive advantages. Within the overall successful IT products and services market, however, ISD projects have long been associated with the "software crisis" and have presented significant challenges to client organisations. In addition to the well-recorded challenges in the literature, the study is also motivated by the author's personal experience of ISD project challenges in the IT industry. The objectives of the research are to understand these challenges and to help client organisations to manage them better. The focus on application software suppliers is due to a combination of the central role of application software in developing successful information systems and the possibility of examining supplier management in ISD projects holistically with well-documented evidence. By focusing on this important area, it is hoped that some contribution can be made to resolving the "software crisis".

Chapter 2: Research Methodology

"The way in which knowledge progresses, and especially our scientific knowledge, is by unjustified (and unjustifiable) anticipations, by guesses, by tentative solutions to our problems, by conjectures. These conjectures are controlled by criticism; that is, by attempted refutations, which include severely critical tests."

(Popper, 1963, p. vii)

"The decision to reject one paradigm is always simultaneously the decision to accept another, and the judgement leading to that decision involves the comparison of both paradigms with nature and with each other."

(Kuhn, 1970, p. 77)

"... I claim that the typical descriptive unit of great scientific achievements is not isolated hypothesis but rather a research programme."

(Lakatos, 1978, p. 4)

"One of my motives for writing *Against Method* was to free people from the tyranny of philosophical obfuscators and abstract concepts such as 'truth', 'reality', or 'objectivity', which narrow people's vision and ways of being in the world. Formulating what I thought were my own attitude and convictions, I unfortunately ended up by introducing concepts of similar rigidity, such as 'democracy', 'tradition', or 'relative truth'. Now that I am aware of it, I wonder how it happened. The urge to explain one's own ideas, not simply, not in a story, but by means of a 'systematic account', is powerful indeed."

(Feyerabend, 1995, pp.179-80).

2.1 Introduction

This chapter first considers the epistemology of scientific research at a philosophical level, outlining the competing schools of thoughts and research approaches in guiding our pursuit of knowledge (Section 2.2). Next, available research strategies are briefly surveyed and a research process is described to unite various research strategies, methods and techniques (Section 2.3). It is suggested that such a research process is fitting for tackling business problems based on the mixed-method research approach. The research strategy and design for this thesis is then mapped out, identifying sources of evidence and the analytic approach (Section 2.4). In Section 2.5, the method of "participant observation with complete membership roles", which is used to collect the first-hand research evidence, is described in some detail. Section 2.6 considers the analytic approach further. The issue of research ethics is considered in Section 2.7. A summary is given at the end of the chapter.

2.2 Philosophical Schools and Research Approaches

There seem to be more philosophical and methodological debates regarding research approaches in social sciences than in physical sciences. These debates are a healthy sign since scepticism "is an essential part of scientific inquiry, and different types of methods represent important critical perspectives" (Brewer and Hunter, 1989, p. 11). In this Section the author attempts to give a brief survey of the mainstream debates gleaned from the literature.

Tashakkori and Teddlie (1998) give an account of four philosophical traditions (mainly based on Lincoln and Guba, 1985 in the first place):

- Positivism, also called logical positivism;
- Postpositivism;
- Constructivism, other variants of which are known as interpretivism, naturalism;
- Pragmatism.

Each of the philosophical schools has its associated "axioms" in terms of ontology, epistemology, axiology, generalisations and causal linkages. For example, positivism has the following "axioms":

- Ontology (nature of reality): positivists believe that there is a single reality (naïve realism);
- Epistemology (the relationship of the knower to the known): positivists believe that the knower and the known are independent (dualist/objectionist);
- Axiology (role of values in inquiry): positivists believe that the inquiry is value-free;
- Generalisations: positivists believe that time- and context-free generalisations are possible;
- Causal linkage: Positivists believe that there are real causes that are temporally precedent to or simultaneous with effects.

According to Tashakkori and Teddlie (1998, p. 7), logical positivism was discredited as a philosophy of science after World War II due to "dissatisfaction" with these axioms. As a result, postpositivism came in its place, with some of the tenets modified from those of positivism (Guba and Lincoln 1998, p. 203 and p. 205):

- Ontology: critical realism. Reality is assumed to exist but only to be imperfectly apprehendable because of basically flawed human intellectual mechanisms and the fundamentally intractable nature of phenomena;
- Epistemology: modified dualist/objectionist; critical tradition/community; findings probably true;
- Methodology: modified experimental/manipulative; critical multiplism; falsification of hypotheses; may include qualitative methods.

Postpositivism remains "within essentially the [same] set of basic beliefs" as positivism (Guba and Lincoln 1998, p. 202). Terms like "positivism", "positivist" and "positivistic" are often used to refer to a series of related philosophical inclinations including positivism and postpositivism. For example, Remenyi et al. (1998) classify research approaches into "positivist" and "non-positivist (phenomenology)".

Constructivism together with the other variants like interpretivism and naturalism is a more radical reaction to positivism. However, unlike postpositivism, which is a direct heir to positivism, interpretivism can be traced back to the "Chicago School" of the 1920s (Hammersley 1989, Denzin and Lincoln, 1998). In contrast to positivism/postpositivism, constructivism/interpretivism adopts directly opposing philosophical positions:

- Ontology: there are multiple, constructed realities (relativism);
- Epistemology: the knower and the known are inseparable;
- Axiology: inquiry is value-bound;
- Generalisations: time and context-free generalisations are not possible;
- Causal linkages: it is impossible to distinguish causes from effects.

In addition to Guba and Lincoln's polarising depiction of these different philosophical schools, Tashakkori and Teddlie (1998) attempt to supplement these attributes with an additional dimension on logic. Thus the positivists are associated with deductive logic, "...an emphasis on arguing from the general to the particular, or an emphasis on *a priori* hypothesis (or theory)" (p. 7). The constructivists, on the other hand, are associated with inductive logic, "...an emphasis on arguing from the particular to the general, or an emphasis on "grounded" theory" (p. 10). However, it has to be pointed out that this added dimension seems contradictory with Guba and Lincoln's assertion that constructivists do not believe in generalisations in the first place.

Associated with different philosophical schools are different sets of research practices, namely quantitative and qualitative research approaches (or research paradigms). These paradigms have been differentiated in the following ways (condensed based on Becker's work as quoted in Denzin and Lincoln 1998, pp. 8-11; Denzin and Lincoln, 2000, pp. 9-10):

- Different underlying philosophical school: positivism/postpositivism for quantitative research and interpretivism/constructivism for qualitative research;
- Acceptance of post-modern sensibilities: quantitative research emphasizes objectivity as a research evaluation criterion while qualitative research believes in multitude of criteria "including verisimilitude, emotionality, personal responsibility, an ethic of caring, political praxis ...";
- Capturing the individual's point of view: quantitative researchers think that empirical materials obtained through softer, interpretative methods (like interviewing and observation) are "unreliable, impressionistic, and not objective";
- Examining the constraints of everyday life: Quantitative researchers "see a nomothetic or etic science based on probabilities derived from the study of large numbers of randomly selected cases. These kinds of statements stand above and

outside the constraints of everyday life. Qualitative researchers are committed to an emic, idiographic, case-based position, which directs their attention to the specifics of particular cases";

- Securing rich descriptions: "Qualitative researchers believe that rich descriptions of the social world are valuable, whereas quantitative researchers ... are less concerned with such details".

In other words, the two different research paradigms prescribe for different types of data, different ways of data collecting, analysis and interpretation, and different representation of research results. That they appear to be totally incompatible seems obvious when one considers the opposite underlying philosophical inclinations. The polarising depictions of the two paradigms have been termed as "the incompatibility thesis". Consequently, "paradigm wars" broke out in the 1980s across several "battlefields" concerning different issues, such as the "nature of reality" and "the possibility of causal linkages" (Tashakkori and Tedlie, 1998).

While the methodology purists were hotly debating "who is right" based on the incompatibility thesis, there were calls for a truce. A compatibility thesis was proposed by Reichardt and Rallis in 1994 (quoted in Tashakkori and Teddlie, 1998) to unite the two seemingly opposite research paradigms. They suggested that the two research paradigm camps shared the following fundamental values:

- Belief in the value-ladenness of inquiry;
- Belief in the theory-ladenness of facts;
- Belief that reality is multiple and constructed;
- Belief in the fallibility of knowledge;

Belief in the underdetermination of theory by facts, that is to say, any given set of data can be explained by many theories.

The advocates of the compatibility thesis suggest a mixed-method approach and adopt pragmatism as the underpinning philosophy. Pragmatism, as a philosophical school, has been

championed by American philosophers like C. S. Peirce, W. James and John Dewey. The central tenet is "whatever works!". One key element of pragmatism is the emphasis on the *application* of the derived knowledge (Field, 2001). The mixed-method research paradigm encourages a combination of qualitative and quantitative methods, each having its own techniques and serving its unique purpose in the process of inquiry. Lincoln and Guba (2000), two of the original proponents of the "incompatibility thesis", acknowledge that:

> "Indeed, the various paradigms are beginning to "interbreed" such that two theorists previously thought to be irreconcilable conflict may now appear, under a different theoretical rubric, to be informing one another's arguments." (p. 164)

The trend in recent years is to use a problem-focused methodology with different methods complementing each other when appropriate (Nau, 1995; Davison, 1998; Kelle, 2001). As Tashakkori and Teddlie (1998, p. 21) state: "...we believe that pragmatists consider the research question to be more important than either the method they use or the world view that is supposed to underlie the method."

It may still be useful to differentiate epistemological research frameworks, one being positivistic and the other being non-positivistic or phenomenological (Remenyi et al., 1998). The strict association of positivistic research to quantitative studies and that of non-positivistic research to qualitative studies, however, prove to be less than useful. As Trochim (2002) points out, "*All quantitative data is based upon qualitative judgements; and all qualitative data can be described and manipulated numerically.*" (emphasis in the original). With qualitative and quantitative research approaches reconciled as a mixed method under pragmatism, consideration is turned next to the research process and research methods that are available to researchers, especially in business and management research.

2.3 Research Strategy and Research Process

Business and management research involves the application and development of theoretical and empirical knowledge from established disciplines to business and management activities

(Wass and Wells, 1994). As such, it is often multi-disciplinary in nature. In addition, the field of study is broad and approaches are wide-ranging (Wass and Wells, 1994; Remenyi et al., 1998). More importantly, there is a strong emphasis on the application of knowledge rather than on the creation of knowledge for its own sake (Remenyi et al., 1998). This study is an example, using a number of concepts and techniques from economic theories with the objective to find a potential solution to ISD project challenges. This section first considers briefly common research strategies for management studies and then discusses a research process that unites different research strategies and methods for tackling business problems.

2.3.1 Research Strategy

There are a variety of research strategies available for management studies. The most common ones are experiments, surveys, case studies and action research (Galliers, 1985; Bennett, 1991; Robson, 1993; Saunders et al., 2000). These strategies are not mutually exclusive and each uses methods and techniques that might be found in others. These strategies are briefly considered here and the selection of the research strategy for this study is considered in Section 2.4.

Experiments (laboratory or field) are common in the research of physical sciences. It has also been used extensively in some branches of social sciences like psychology (Saunders et al., 2000). The researchers are expected to isolate and control variables in experiments (Galliers, 1985). Due to complex social and organisational constraints, experiments are often not appropriate for researching management and organisational issues (Bennett, 1991). However, the concept of experimentation has played a role in developing the action research method, which will be discussed later in this section.

The survey strategy is popular in business and management studies. The main methods used in surveys include questionnaires and interviews (Robson, 1993). The researcher may use

mail, telephone or face-to-face meetings to reach survey targets, each with its advantages and disadvantages (Jobber, 1991). For example, the mail survey may be economical but requires the survey questions to be highly structured. On the other hand, the face-to-face survey allows a researcher to take a more flexible approach, though much more resource-intensive. The common characteristic of the survey method is that the questions posed by the researcher are structured. In surveys, what kind of information to be obtained can be predicted, and extra information is often discouraged (Becker, 1996). This makes the survey strategy inappropriate for researching complex and open issues.

The case study strategy has often been presented in contrast to survey in management studies (Smith, 1991). It refers to a fairly intensive examination of a single "case" or a small number of related "cases". The approach allows the researcher to generate answers to the question "why" as well as "what" and "how" (Robson, 1993). Case studies may incorporate a range of different research methods, including observations and interviews. Stake (1994) suggests "case study is not a methodological choice, but a choice of object to be studied". This thesis is based on the case study strategy. The "object to be studied" is ISD projects involving external application software suppliers (see Section 2.4 for research design details).

Action research (AR) might be considered as a subset of case studies, but its underlying philosophy sets it apart (Galliers, 1985). The AR strategy intends to introduce certain changes into a real social context and study the resultant effects. AR does not purport to control all variables, thus it is different from the experiment research strategy discussed above. One prerequisite for this strategy is some existing theoretical framework for the researcher to formulate the intended change in the first place. This makes the strategy useful for "theory elaboration and development" (Eden and Huxham, 1996) but not for theory generation. In other words, AR might be more suited for theory-testing than theory-building.

Research strategies may also be classified according to the purpose of the inquiry into that of descriptive, exploratory and explanatory (Robson, 1993; Yin, 1994). The choice of the research strategy may depend on many factors, not least the research questions and access to research data etc. The choice of research strategy for this thesis is considered in Section 2.4.

2.3.2 Research Process

Research process is usually depicted to start from research planning and literature review or evidence collection and to end with a theory or a research report with management guidelines/recommendations (Wass and Wells, 1994; Remenyi et al., 1998; Day, 1998; Flick, 1998). Such a description may be appropriate within certain contexts. However, they give the impression that the research process start and finish with one individual researcher without involving corresponding research communities. In addition, they do not reflect the complex nature of applying research results to real world problems. This section considers a possible research process framework that can accommodate wide-ranging concepts from various research approaches. The framework emphasises the objective of management research as solving substantive business problems. Moreover, the framework recognises a spiral improvement process along which a theory can be developed, applied and refined. This is perhaps what Lakatos (1978) means by a "research programme". Lakatos disagrees with Popper (1959, 1963) in that scientific researches are based on simple hypothecation and falsification. Rather researchers develop research programmes to deal with anomalies incompatible with the existing hypothesis and theories on a longer-term basis.

The process framework is shown in Figure 2.1 and is proposed based on the following observations:

- Real problems in the business world are complex and unwieldy, and are usually beyond the means of a single research project (see e.g. Benbasat and Zmud, 1999). Division of labour is therefore often a necessity;

- Researchers depend on their peers heavily for discussions and constructive criticism;
- Business entities are risk-averse, therefore application of any new potential solution to any substantive problem can and should only happen after careful risk-assessment by the research community (via peer-reviews) and the business stakeholders (e.g. cost-benefit analysis);
- Various research methods and techniques are best understood within a unified research process. This helps to avoid developing each technique into a separate "methodology".

Each stage of the process is discussed below with appropriate methods, always with business research in mind.

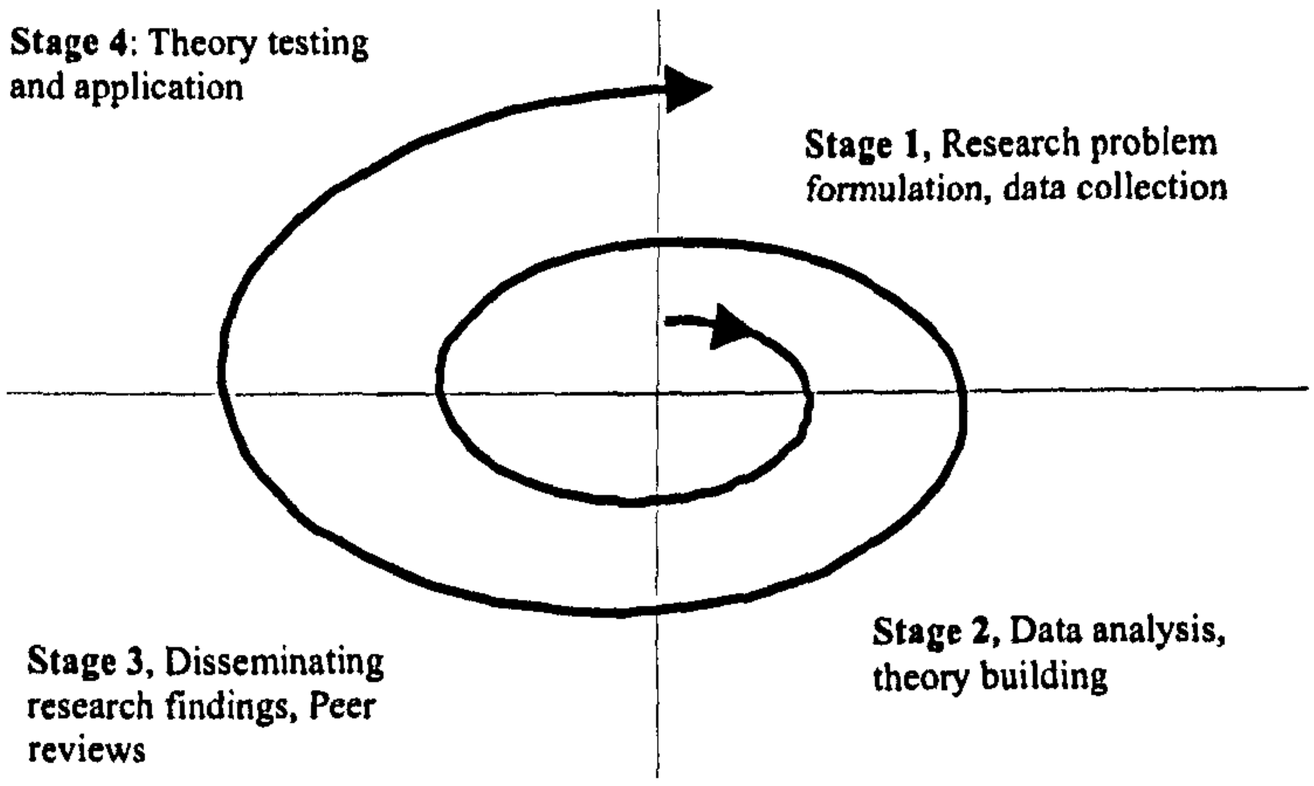


Figure 2.1: Research process spiral

Stage 1: Research Problem Formulation and Data Collection

Research problems are formulated and data collection is carried out at this stage. This stage might be further divided into different phases. Like many other activities in research, problem formulation and data collection tend to intertwine (for a discussion on this issue see Brewer and Hunter, 1989). There are a variety of research methods suitable for this stage. Broadly speaking, they can be grouped into observing, interviewing and questionnaires (Robson, 1993). Observational methods might be further classified into formal or informal approaches according to how structured the information gathering process is. The formal observation is

more structured while the informal approach allows unstructured and complex information to be gathered. Another related dimension is the role adopted by the researcher. Thus, the researcher may be a pure observer with little participation at one extreme to a complete participant (Adler and Adler, 1987; Robson, 1993).

Interviews and questionnaires are "self-report" techniques since they rely on respondents' revealing about themselves (Robson, 1993). They are the main methods used in survey (Jobber, 1991; Section 2.3.1), though they can be used in combination with other strategies like case studies. Other data collection methods include archival analysis and simulation etc (Robson, 1993; Yin 1994). For this study, the author adopts a complete participant approach for data collection (see Section 2.4 for the research design for this study).

Stage 2: Data Analysis

While Stage 1 in Figure 2.1 is descriptive, Stage 2 is analytical. Writing features highly in this stage. Writing not only helps remembering observations, sources and ideas, it induces thinking and improves understanding (Booth et al., 1995). That said, writing is part of thinking and naturally accompanies any stage of research, not separable, for example, from evidence gathering (Denzin and Lincoln, 1998, pp. 19-21). For some research questions, research stops at Stage 1 (called descriptive research by Trochim, 2002 or fact-finding research by Brewer and Hunter, 1989). Most problem-solving researches, however, require further effort to uncover correlational links or causal links among variables.

The traditional tool employed for data analysis is induction. Grounded theory is one of the methods conceived to serve such a purpose (Glaser and Strauss, 1967). It is a "qualitative research method that uses a systematic set of procedures to develop an inductively derived theory about a phenomenon" (Strauss and Corbin, 1990, p. 24). Grounded theory was proposed in the 1960s as a reaction to "grand theoretical schemes" prevalent in the mid-1900s

(Locke, 2000, p. 26). The method provides codified procedures for data analysis consisting of note-taking, coding, memo-ing, sorting and writing (Dick, 2000). Despite the claim made about grounded theory being a general method of analysis (Lowe, 1996), the specialised terminology and procedures mean that its use is limited to certain types of data. What is more problematic is a certain amount of confusion within the proponents of grounded theory. While Strauss (1987) argues for all modes of activity – induction, deduction and verification – to enter into the process of inquiry, Orlikowski (1993) identifies three characteristics of grounded theory method, one of them being specifically inductive (the other two being contextual and processual). Researchers practising grounded theory methodology give conflicting advice regarding whether the method is suitable for new researchers (cf. Urquhart, 2000 and Davidson, 2001). There have been divergent opinions between the two originators regarding the "right" way, with Glaser taking a more "emergent" approach while Strauss is seen to give more prescriptive guidance (Babchuk, 1996; Locke, 2000; Dick, 2000; Smit and Bryant, 2000). Such conflictions indicate that there are perhaps fundamental issues yet to be addressed within the methodology. Smit and Bryant (2000) observe that "recent developments in the method have centred around the split between Glaser and Strauss, rather than on genuine progress and evolution". A full appraisal of grounded theory is beyond the scope of this thesis, but it can form an interesting separate study to examine the methodology critically with the benefit of the epistemological debates outlined in Section 2.2.

Triangulation is another research method often mentioned in literature. According to Tashakkori and Teddlie (1998, p. 18), Campbell and Fiske pioneered the concept in 1959 and "used more than one quantitative method to measure a psychological trait". In 1978, Denzin further "described four different types of triangulation methods, including data triangulation, investigator triangulation, theory triangulation and methodological triangulation." Triangulation is based on an analogy drawn with the surveying technique. It is a simple concept, as described in Hutchinson Reference (1991):

> "In triangulation, surveyors measure a certain length exactly to provide a base line. From each end of this line they then measure the angle to a distant point, using a theodolite. They now have a triangle in which they know the length of one side and the two adjacent angles. By simple trigonometry they can work out the lengths of the other two sides." (p. 834)

In other words, triangulation in surveying is based on trigonometry to determine the distance between a surveyor and a far-away point. If the surveyor carries out the readings correctly and the instruments used are accurate, the results will be accurate within the expected tolerance level. As a research method, however, triangulation does not give us any certainty to match what surveyors enjoy. In using triangulation, the researcher merely gains different perspectives. There is no inherent equivalent research trigonometry to guide the researcher to reach a conclusion. The real test of theories and hypothesis is at a later stage of application. Triangulation may enhance the researcher's confidence in the facts and evidence obtained but it does not provide any *certainty*. It may not replace application or falsification as part of the overall research process.

Stage 3: Research Result Dissemination and Peer Reviews

Stage 1 and 2 are recognised in conventional terms to constitute a research process (e.g. Saunders et al., 2000; Remenyi et al., 1998). Some types of research stop here, especially those commissioned by organisations where research results are for internal consumption only and not visible to wider academic communities. However, many research results do find their way into academic communities via seminars, journals, books, the Internet and other channels. Peer-reviews in these channels are part of the research process. Due to the underdetermination of theory by facts, new observations can be afforded different interpretations by other researchers with different critical perspectives.

In business and management research "there is a strong emphasis on the application of knowledge rather than on the creation of knowledge" (Remenyi et al., 1998, p. 10). This means that not only academic researchers, but also practitioners in the business community

will be interested in and be able to provide valuable feedback on relevant research results. If research results are to be applied in business organisations, it becomes critical to get the practitioners involved. This is believed not to be the case currently in some aspects of management studies. Benbasat and Zmud (1999) lament about the lack of relevance of information systems research to practitioners. Whether the lack of relevance is due to research topic selection or is due to the lack of communication between the academic and business communities is worth further inquiring.

It should be recognised that testing new hypothesis in real world organisations, especially on substantive management problems, is inherently risky. A peer-review stage could be construed as part of risk management process. So far as target businesses are concerned, such risk assessment activities are nothing short of essential. Possible scenarios should be mapped out, possible anomalies anticipated and disaster recovery plans thought through.

One thing to guard against, however, is to see practitioners as judges for research results (e.g. Remenyi et al., 1998). Without appropriate training, practitioners lack necessary theoretical understandings and are thus ill-equipped to evaluate research results constructively. Practitioners carry valuable experiences that can shed a different light onto a research problem, but they are not researchers. Mintzberg (1973), when discussing research methods on managerial work, comments:

> "It would appear that the simplest way to find out what managers do is to ask them, by way of interviews or questionnaires... The results have been disappointing...to ask the manager what he does is to make him the researcher; he is expected to translate complex reality into meaningful abstraction. There is no evidence to suggest that managers can do this effectively; in fact there is ample evidence from empirical studies... that managers are poor estimators of their own activities. Despite their convenience, the interview and questionnaire methods should be recognised as useful only in the study of managers' perceptions of their own jobs." (p. 222)

Consider the research problem of finding out the success/failure causes of ISD projects as an example, practitioners may be able to give their *perceptions* of what may have gone wrong,

but they can by no means be expected to uncover the true root causes.

Stage 4: Application of Research Results

Stage 4 is a stage of applying new knowledge and understandings in real world organisations. The application of theory serves two objectives: one is to test the theory from the scientific inquiry point of view and the other is to solve practical problems. One technique that might be useful for this stage is that of action research (AR). To put it simply, AR is to introduce certain changes into a real social context and study the resultant effects. AR does not purport to control all variables, thus it is different from laboratory experiments. There have been increasing attention paid to AR in recent years in ISD research (Kock and Lau, 2001). Kock and Lau (2001) caution on the use of AR to test theories:

> "...one may try to test a model (or a set of hypotheses) by conducting an action research study. The problem with this choice is that the lack of control and narrowly focused data collection, which is typical in action research, will almost always ensure that such tests do not conform to well-established positivist methodological standards."

This caution may be valid when considering the complex nature of research context and the often-conflicting interests of stakeholders within, for example, an ISD project. But if the "well-established positivist methodological standards" refer to those uses in the classical laboratory experiments with all variables controlled, this comparison is misplaced. By definition, AR is different from laboratory experiments. AR, as a research method, can also be used as a starting point to build a theory in combination with participant observation and appropriate analytical methods. A research project designed in this way effectively starts the research cycle from Stage 4, and progresses to Stage 1 and Stage 2 later. This is perfectly valid and logical. It demonstrates that the numbering of stages in Figure 2.1 is for conceptual convenience only. The wheel of learning does not have a fixed starting point.

In summary, a typical business research process might be described as a journey from problems to potential solutions in the shape of hypotheses and theories, then to peer-reviews

of the solutions, usually aiming for the application of the solutions. The application is likely to yield unexpected observations, and anomalies contrary to the original theories. Further observations and evidence require further analysis and thus the research carries on. Dividing the research process into different stages does not mean to compartmentalise activities mechanically. It is recognised that observations and hypothesis building activities are closely interwoven. The described stages are not meant to be prescriptive either. The application of theory (Stage 4) may not happen if a given theory is not mature enough. Figure 2.1 is an effort to unite some of the seemingly disparate research methods for solving business problems. Such a conceptual scheme may be helpful in guiding business and management research toward a general goal of application, thus increasing its relevance without necessarily sacrificing the rigour of inquiry.

2.4 Research Design

Given the research motivation (Section 1.4) and research objectives (Section 1.5), and bearing in mind the research process discussed above, this Section considers the research design for this study. The choice of the research strategy is discussed in Section 2.4.1 and the research components are outlined in Section 2.4.2. Methods for data collection and data analysis are further discussed in Section 2.5 and 2.6.

2.4.1 The Adopted Research Strategy

The focus of this thesis is on identifying the root causes of ISD project challenges and possible courses of remedial actions. There is no pre-existing theoretical framework assumed for the research. Thus the research can be said to be exploratory and explanatory in identifying causality on one hand and assume a problem-solving approach rather than testing any *a priori* theory on the other. To achieve this purpose of inquiry, the adopted research strategy is that of case study (Section 2.3.1). Comparing with other options, the choice is made due to the following reasons:

- ISD projects take place in complex social settings involving many individuals, teams and organisations. Variables and factors affecting ISD projects are not conducive to isolation and control. Therefore, the experiment strategy is ruled out for this study.
- The survey strategy has been used by many previous researchers in examining the root causes of ISD project challenges and failures (Section 3.5). Though surveys results are useful in many ways, the value of repeating the previous studies is not particularly high. In addition, owing to the rigid and structured nature of the survey strategy, it may have contributed to the identification of superficial reasons for project challenges but not the "real" root causes (Section 3.7). Therefore, the survey method is not used for this study, though many of the previous survey results are incorporated later in the thesis.
- The action research (AR) strategy requires an existing theoretical framework. Since the objective of this study is exploratory and aims for theory-building rather than theory-testing, the AR strategy is therefore unsuitable. However, the author recommends the AR strategy for possible follow-up studies (Section 11.5.2).

Within the general case study approach, Yin (1994, p. 80) identifies six sources of evidence:

- Documentation;
- Archival records;
- Interviews;
- Direct observations;
- Participant-observation;
- Physical artefacts.

For this thesis, participant observation with "complete membership role" (Adler and Adler, 1987) from three ISD projects provides the core research evidence supported by project documents, private conversations, emails and software artefacts. The research is based on three ISD projects encountered by the author while working in the IT industry. The projects

have not been selected or "sampled". There was no deliberate change or control introduced into any of them for the purpose of this research. According to Dyer and Wilkins (1991), such an approach is more suited to the exploratory research for the purpose of "theory-building". In contrast, case studies that are based on "sampled" or "controlled" cases are more relevant for theory testing.

With reference to the research process described in Section 2.3.2, the strategy is to limit the research to recording empirical evidence, formulating and refining a hypothesis and proposing a potential solution. These activities fall within Stage 1 and Stage 2 of an initial research spiral (Figure 2.1), leaving the application (or falsification) of the derived proposition to separate studies after a suitable peer-review stage. Such a strategy carries with it inherent limitations, the major one being the lack of empirical hypothesis/theory testing. This is based on two considerations. The first is the recognition of the magnitude of ISD challenges, and the limited resources and time available for a PhD project. To carry out both theory-building and theory-testing within such limited resources may lead to "premature empirical testing" (Haig, 1995). The second consideration is that meaningful empirical testing of the proposed solution in real world organisations should only take place after careful assessment of risks and the associated costs and benefits to the target organisations (Section 2.3.2). In addition, the theory of scientific inquiry consistently reminds us that it is only possible to verify propositions by elimination and not possible to "prove" a proposition by confirmation (Popper, 1959, 1963; Kuhn, 1970). What is important is to establish propositions that are falsifiable and to reveal, as far as possible, the assumptions inherent in the explanations (Bakker and Clark, 1988).

2.4.2 Research Components

Yin (1994) suggests to consider the following research "components" in designing case studies:

- Study questions;
- Study propositions;
- Unit of analysis;
- The logic linking data to propositions;
- The criteria for interpreting the findings.

These components are discussed below relating to this study.

Study Questions

The study questions are based on the research motivation (Section 1.4) but slightly expanded.

Two central questions are addressed:

- What are the root causes of ISD project challenges?
- What can be done to address the root causes?

There are other related questions. In particular, before the first question can be answered, it is important to understand the project challenges in the first place. In this respect, the case studies based on participant observation help significantly due to the understandings of project background, associated contracting parties and their relationships. In addition, it is important to understand the current explanations of the challenges and available solutions and why they may not have been effective in overcoming the project challenges.

Study Propositions

A proposition "directs attention to something that should be examined within the scope of the study" (Yin, 1994, p. 21). In that sense, the central proposition for this study is that the management of application software supplier is the key to the successful delivery of quality application software, which in turn is the key to the success of an ISD project (see Section 1.6). The study proposition is not a foregone conclusion that cannot be disputed. Rather, it provides a basis for data collection and helps to limit the scope of the study (Yin, 1994).

Unit of Analysis

Three projects are documented in this thesis as three case studies (Chapters 4, 5 and 6). However, the main unit of analysis is one client-supplier relationship since the research proposition is related to supplier management. In Case 1, two such relationships are described while Case 2 and Case 3 each describes one. Each client-supplier relationship is established within the context of the corresponding project. The "time boundaries" (Yin, 1994, p. 24) for data collection are limited to the successive phases of supplier selection and contracting, software product development, delivery, evaluation and integration, not including system installation and system support. While the client-supplier relationship forms the basic unit of analysis, some issues are discussed at the project level when necessary.

The Logic Linking Data to Propositions, and Criteria for Interpreting the Findings

Yin (1994) admits that these two components, representing data analysis steps, have been "the least well developed in case studies" (p. 25). While Yin does not give much general guidance regarding these components when discussing research design, he does provide useful suggestions in a later chapter about analysing case study evidence (Yin, 1994, pp. 102-125). A number of analytic techniques are suggested. Two of them are used in this study. One is "pattern-matching" (Yin, 1994, p. 106). Applying this technique, a pattern of ISD project challenges is derived in Chapter 7 based on individual cases. This pattern is then compared with what has been described in literature. The other technique adopted is "explanation-building". This is central to the study since one of the objectives is to find the root causes of ISD project challenges. The initial explanations are based on a number of existing theoretical concepts found in the literature of product development, agency theory and contract theory etc (Chapter 8). The analysis of the research evidence produces a generalisation in the form of a unified explanation. In more traditional research methodology parlance, the process is described as "analytic induction" rather than "enumerative induction" (Znaniecki as quoted in Robinson, 2000) or "analytic generalisation" rather than "statistical generalisation" (Yin,

1994, p. 30). Ideally, a set of pre-determined criteria should be specified to examine the initial explanations and generalisations. However, according to Yin (1994, p. 26), currently "there is no precise way of setting the criteria for interpreting these types of findings".

The guidance available in the literature about making generalisation in case studies has been limited. Benbasat et al. (1987) resort to "the integrative powers of the researcher". Eisenhardt (1989b) prescribes a series of steps involving pre-defined "constructs", which might be more relevant to "theory-testing" rather than exploratory "theory-building" studies (Dyer and Wilkins, 1991). However, merely maintaining that theories will emerge from research data (e.g. Glaser and Strauss, 1967) without being explicit of the reasoning process does not help to increase the rigour of research methodology, nor does it allow peer researchers to offer effective critiques of research findings. In the meantime, the author recognises that the epistemological process has been addressed as a general philosophical issue. Instead of formulating a special process for making generalisation from case studies, it might be appropriate to seek advice available in the epistemological literature. One source of advice is found in the "Inference to the Best Explanation (IBE)" process described by Lipton (1991). IBE is adopted as the overall reasoning framework for this study incorporating some of the specific research techniques for case studies described earlier. For a detailed discussion of the IBE process as applied in this thesis, see Section 2.6.

2.5 Data Collection Based on Participant Observation

The core evidence of this study comes from three first-hand case studies based on participant observation with complete membership roles (Adler and Adler, 1987). The author was in employment from May 1999 to February 2003 with a medium-sized information systems "solution provider" company, coded as SLC. From July 2000 to March 2002, the author participated in three ISD projects that have been documented as case studies in Chapters 4 to 6 (also see Section 1.4). The ethical issues are considered in Section 2.7. This Section

discusses participant observation as a research method and provides further background information for the cases documented.

2.5.1 Participant Observation as a Research Method

Participant observation as a research method has its roots in the "field work" of the Chicago Schools in the 1920s. By the 1950s and 1960s, the technique had been further developed and codified by the so-called second generation of Chicagoan social scientists represented by Herbert Blumer, H.S.Becker and Everett Hughes (Adler and Adler, 1987). According to Adler and Adler, participant observation has the following characteristics:

- The goal is to contribute to general theoretical statements about cultural and social life;
- In order to do this, it is necessary to venture, first-hand, into the places where the activities of interest are taking place, and to observe human group life "*in situ*";
- Researchers can take various roles including complete observer, observer-as-participant, participant-as-observer and complete participant, ranging from least to most involved in the group and its activities.

The strengths and weaknesses of the approach as used for this study are discussed in the next section.

2.5.2 Participation in the Case Study Projects

Historically, SLC used to be the Information Systems Division of UKOne. As a division, SLC had always undertaken the development of many information systems in support of the business operations of UKOne's various subsidiaries. The systems were usually developed jointly with other software suppliers with specific business domain expertise and technological know-how. As of April 2000, SLC was spun off as a separate company with a new identity. Part of the separation agreement was that UKOne had guaranteed a level of workload over five years for SLC. This was through two type of activities, one being existing

systems' support and maintenance, the other being new systems development, usually in collaboration with other suppliers for UKOne's the subsidiaries.

The three projects described in this study were undertaken by UKOne's three different subsidiaries. It is worthwhile to note that:

- Cases are not selected because of any particular displayed feature. Rather they are "random" in that they have happened to be within the author's experience on a continuous basis from July 2000 to March 2002;
- Each case does have its own distinctive client. Each subsidiary has its own personnel, its own locale, its own approach to ISD projects and its own sub-culture. It means that the systematic biases associated with single-location case studies are under check even within the limited time period and a single overall corporate entity.

The case studies have been written after the projects had ended. This approach has a number of implications. First of all, the approach seems to have avoided two issues traditionally associated with it when the researcher is a complete participant. One is the risk of "going native", the other the need to take a "covert" strategy. Being a "native" to start with, the author did not have a separate agenda or an "outsider" view (Spradley, 1980). At the time of carrying out the project duties, the author did his job with the best of his ability in accordance with company procedures. There was no need to take any "covert" strategy. Secondly, the observations thus acquired in the day-to-day project involvement are unstructured. Unstructured observation has been used and has proven as an effective method to deal with the complexity of a research situation. Mintzberg (1973) assigns two "great strengths" to this method:

> "First, the researcher can be purely inductive. He imposes on himself no artificial constraints or premeditated structure. Nothing need stand between the work he observes and the theory he develops save his own ability to interpret. Second, the researcher is in a position to probe deeply into the work that he observes. If an incident is unclear, he is there and he has the time to find out more." (p. 226)

The second strength is only relative to the other research methods like survey. In surveys, what kind of information to be obtained can be predicted, and extra information is often discouraged (Becker, 1996). Unstructured observation does not give the researcher any artificial limit in terms of the types and amount of information. The researcher is out there to gather everything possible that is of interest, including "a mixture of events, anecdotes, views and attitudes of those observed, documentary evidence, and so on." (Mintzberg, 1973, p. 226). This is essential for a creative, and open-ended research.

The downside of the unstructured observation is its subjectivity. Although subjectivity is recognised as a universal issue in many qualitative methods in social inquiries, it can be potentially more significant with unstructured observation. The author relies on available project documentation and artefacts for accuracy check at the time of writing up the cases rather than purely relying on personal observations. Typical ISD projects generate voluminous documentation from requirement specifications, software design and architecture, testing specifications, testing evidence and various project reports. Artefacts are mainly software codes and patches, software installation instructions and user guides etc. Careful crosschecking of research evidence helps enhance the reliability of the case materials but it does not eliminate the subjectivity all together. The observations available for analysis represent the author's perceptions of ISD projects under study. That such perceptions can be rooted in realities is a fundamental belief of critical realists and provide the basis for further epistemological inquiries.

2.5.3 The Structure of Case Study Composition

Yin (1994, pp. 136-141) identifies six typical structures for presenting a single case study:

- Linear-analytic;
- Comparative;

- Chronological;
- Theory building;
- "Suspense";
- Unsequenced.

The "linear-analytic" structure is adopted for presenting the case materials in Chapters 4, 5 and 6. Such a structure consists of a "sequence of subtopics" (Yin, 1994, p. 138) relevant to the case study. Each of the three chapters reports on one project. Each case is generally made up of two parts. The first part is descriptive including the project background, the client and its requirements, the selection of suppliers and contract arrangement, the management of project schedule, cost and quality. The second part presents a within-case analysis that focuses on issues in each project based on observation. This part of the case studies is less structured than the first part since each case tends to have different issues. In addition, each case study includes a project chronology and a section of internal references.

2.6 From Observation to Understanding

The empirical inquiry starts from observations and facts (in a critical realistic sense). But what is the process that leads us from what we observe to a level of general understanding? And how do we assess the validity of our understanding? Yin (1994) acknowledges that these two areas are "the least well developed in case studies" (Section 2.4.2). Of course, these questions are not limited to the case study research methodology. Rather they are general epistemological issues. Traditionally induction is considered to be the process to generate new understandings. However, there are problems with induction as an epistemological process. One problem is that the observed data does not justify the unobserved. When we reach a conclusion that "all ravens are black", what we mean is the conjunction of "all observed ravens are black" and "all unobserved ravens are black". There is no empirical justification for the second statement (Psillos, 1996). Secondly, the argument for the validity of induction is circular. These are two problems underlying David Hume's sceptical argument about

induction (Lipton, 1991). The problems are so serious that Popper (1959, 1963) proposes to abandon induction altogether, opting instead for "conjectures" to generate hypotheses and then relying on refutations to falsify them.

However, Hume himself did not conclude that inductive reasoning should be rejected (Rosen, 1999). Instead, the problems of induction make us aware of the fallible nature of our understanding and the need for supplementary techniques. The "Inference to the Best Explanation (IBE)" process (Lipton, 1991), also referred to as the abductive reasoning (Josephson, 1996, 2001; Psillos, 1996), promises to overcome the problems associated with induction. This next section examines the basis of IBE followed by a section about the criteria of "the best" and an overview of the process as applied in this study.

2.6.1 Inference to the Best Explanation (IBE)

Josephson (2001) equates IBE with the abductive reasoning and defines it as a form of inference that follows a pattern like this:

> "D is a collection of data (facts, observations, givens).
> [Hypothesis] H explains D (would, if true, explain D).
> No other hypothesis explains D as well as H.
> ---
> Therefore, H is probably correct." (p. 1622)

The strength of the conclusion thus reached, according to Josephson (2001), depends on:

- how decisively the leading hypothesis surpasses the alternatives;
- how well the hypothesis stands by itself, independently of the alternatives;
- how thorough the search was for alternative explanations.

Psillos (1996) sets down four desiderata for a general model of ampliative reasoning and suggests that abductive reasoning meets all four. First of all, abduction produces "the best explanation", which can be defeated when an even better explanation is found (it is non-

monotonic). Secondly, abduction warrants generalisations due to its explanatory considerations (it solves the "cut-off" problem"). That is to say, abductive generalisations are more than simple enumerative correlations. Thirdly, abduction "employs hypothesis that involve reference to unobservables and need vocabulary that goes beyond the conceptual resources of the evidence" (it caters for "vertical" extrapolation). Lastly but most importantly, abduction accommodates the eliminative dimension of ampliative reasoning. That is to say that abduction considers competing potential explanations and eliminates all but one of them.

In contrast, induction, in its simple enumerative form, only accommodates the first of the above desiderata (Psillos, 1996):

> "Enumerative induction can accommodate only the first of the above: it is clearly non-monotonic. For instance, if added in the premises, the information that the sample was biased undercuts the previous conclusion. But it fails on the rest. It cannot solve the 'cut-off point' problem. It does not tell us when the evidence is enough to warrant the generalisation. Without extra information about the quality of the data, we cannot tell when we have seen enough instances to warrant the generalisation. It cannot accommodate 'vertical' extrapolation. In a certain sense, enumerative induction, pretty much like deduction, operates with the principle 'garbage in, garbage out': the descriptive vocabulary of the conclusion cannot be different from that of the premises. Hence with enumerative induction, although we may arguably gain knowledge of hitherto unobserved correlations between instances of the attributes involved, we cannot gain 'novel' knowledge, i.e., knowledge of entities and causes that operate behind the phenomena. Finally, enumerative induction cannot accommodate the eliminative dimension of ampliative reasoning: unless we consider alternative generalisations of the observed correlations, we cannot attempt to eliminate them. But enumerative induction does not have the resources to generate alternative generalisations."

However, induction may be considered as a special case of abduction (Harman, 1965; Josephson, 1996; Psillos, 1996). It seems that on a philosophical level, abduction or IBE is a superior mode of ampliative reasoning which overcomes, to a considerable extent, the scepticism resulted from the enumerative induction. The concepts of abduction and enumerative induction coincide with the analytic and statistical generalisations often discussed in research methodology literature (Yin, 1994; Robinson, 2000). Looking in this light, one can appreciate the reason that Yin (1994, p. 31) refers analytic generalisation to be higher (theory level) than that of statistical generalisation (population level).

2.6.2 Criteria of "The Best" Explanation

An important question to be addressed is what is meant by "the best" in the phrase "the best explanation". Thagard (1978) lists three criteria: consilience, simplicity and analogy. Consilience measures how much a theory explains. Thagard (1978) states:

> "Roughly, a theory is said to be consilient if it explains at least two classes of facts. Then one theory is more consilient than another if it explains more classes of facts than the other does. Intuitively, we show one theory to be more consilient than another by pointing to a class or classes of facts which it explains but which the other theory does not." (p. 79)

A number of other terms have been used to denote the concept of consilience, including "explanatory power", "systematic power" and "unification" etc, though they may have different meanings depending on their exact definitions (Thagard, 1978; Lipton, 1991). A theory can be highly consilient if it is allowed to be complicated with many ad hoc assumptions and "auxiliary hypotheses". The criterion of simplicity therefore puts a constraint on consilience: "a *simple* consilient theory not only must explain a range of facts; it must explain those facts without making a host of assumptions with narrow application." (Thagard, 1978, p. 87, emphasis in the original). The third criterion is analogy. Thagard argues that analogies support theories by improving the explanations that the theories are used to give" (Thagard, 1978, p. 89). Of the three criteria, consilience and simplicity are widely recognised. As for analogy, it is not clear how it acts as a criterion to judge how good an explanation is. Analogy may be considered more as a technique to assist formulating and defending hypotheses (Section 2.6.3).

Lipton (1991) distinguishes between the "likeliest" explanation (the one that best fits the available evidence) and the "loveliest" explanation (the one that, if true, would provide the most understanding). "Likeliness speaks of truth, loveliness of potential understanding" (Lipton, 1991, p. 61). These are clearly two different standards. The likeliest explanation may not be enlightening. Johnson (2000) provides the following example. The fact someone is

elected as a governor can be convincingly explained by one's numerical majority in an election. But such an explanation does not tell us anything about the candidate's policy stances or any other information. Therefore the explanation is not lovely. On the other hand, an explanation may be lovely without being likely, like some conspiracy theories (Lipton, 1991). Lipton argues that the criterion of loveliness is more fundamental to IBE. This is because "...the explanation that would, if true, provide the deepest understanding is the explanation that is likeliest to be true" (Lipton, 1991, p. 63).

2.6.3 The IBE Process and Its Application in This Study

IBE provides a structure for reasoning and deriving "warranted inferences" (Harman, 1965; Lipton, 1991; Johnson, 2000). It suggests a two-stage process. The first stage is the generation of the best potential explanation, equivalent to the conventional hypothesis. The second stage is the selection of the best explanation. The first stage relies on background knowledge and guesses. This is in line with the conjecture school of philosophy of science (Popper, 1959, 1963). It is also in line with the scientific tradition that researchers are required to be familiar with their field of research and literature. The emphasis on background knowledge is in contrast with advice from grounded theory methodology that a researcher should be "avoiding the literature most closely related to what you are researching" (Dick, 2000). IBE does not have any guard against any existing literature.

A new potential explanation compatible with available evidence is not enough to be justified as "the best" explanation. IBE suggests a "contrastive inference" process that puts the new potential explanation to the test through a fact-foil analytical process (Lipton, 1991). This enriches the explanation and can provide new understanding. In addition to contrastive analysis, analogies might be used to draw similarities between different phenomena to support explanations and suggest possible new hypotheses and guide new conjectures. Analogical reasoning is a technique widely used in science (Dunbar, 1995; Holyoak and Thagard, 1997).

Dunbar (1995) suggests that "analogies were an important source of knowledge and conceptual change". The exact process of analogical reasoning is a separate topic of research (e.g. Blanchette and Dunbar, 2000) that will not be discussed further in this thesis.

This research follows the IBE reasoning structure. Three case studies (Chapters 4, 5 and 6) based on participant observation are documented with within-case analyses. Chapter 7 provides the first part of the cross-case analysis that shows a pattern of ISD project challenges. This pattern is derived not only from the cases recorded in this thesis, but also incorporating evidence from many published case studies. In particular, this pattern matches similar descriptions of ISD project challenges in the literature (e.g. Yourdon, 2002; see Section 7.7). Chapter 8 provides the second part of the cross-case analysis, which attempts to explain the observed pattern. With reference to concepts and theories in agency theory, transaction cost theory, contract theory and product development literature, and combined with the author's background knowledge and literature survey, the author formulates a new potential explanation by constantly comparing with alternative explanations. The new potential explanation is subjected to two instances of contrastive analysis (Chapter 9). The contrastive analyses are conducted through a fact-foil structure and "causal triangulation" (Lipton, 1991, p. 54) to refine the explanation while the analogical analyses are carried out by examining practices in comparable situations. It is based on the refined explanation and suggestions from the analogical reasoning that a management framework is formulated as a possible solution (Chapter 10).

In summary, IBE provides guidance for exploratory and explanatory studies. Though the criteria of "the best" may appear to be inevitably subjective, appealing too much to intuition, IBE does represent advancement from pure "conjectures". Indeed, one important benefit of IBE is to lay bare the process of reasoning or the "intermediate lemmas" (Harman, 1965). Josephson (2001) suggests that:

> "It is desirable to adopt reasoning strategies that produce conclusions that can in principle be justified. Even better are reasoning strategies in which the confidence in a conclusion arrives bearing a structure of argumentation that can be critically examined, and in which little or no additional work is required to extract, organise, or economize arguments."

In the mean time, it has to be added that IBE as a reasoning framework is still in the process of being developed and refined. In some small way, this study may contribute to the development of IBE by offering an example of its usage.

2.7 Research Ethics

Business research is conducted through observing and analysing business activities of individuals and organisations, and thus has the potential of infringing the interests of those involved. Therefore, the research activities must observe relevant ethical principles. These principles may not always fit with each other in a given context, hence there may also be a need to make a considered choice for a specific approach, be it at the research topic selection, data collection or other stages. One principle is that a research project should benefit society (SRA, 2002; Remenyi, 1998). SRA (2002) puts it succinctly:

> "Social researchers should use the possibilities open to them to extend the scope of social enquiry and communicate their findings, for the benefit of the widest possible community." (Section 1.1)

ISD projects consume significant sums of economic resources, and a considerable portion of which is wasted (Section 1.4). In addition, the lack of ISD project success exerts adverse financial and psychological pressures on organisations and individuals involved. It is in the society's interest to have the workings of these projects examined from different perspectives so that plausible ways can be found to reduce the economic wastes and to improve the organisational and psychological welfare for those adversely affected.

Having said that, there are ethical questions to be addressed. One key question concerning this research is that whether the participant observation and the project documentation should be used on an "informed consent" basis or not. It is a difficult decision to make, particularly

difficult for an employee researcher (SRA, 2002; Buchanan et al., 1999). Informed consent in principle is the ideal approach. It will relieve the researcher of any uncertain and perhaps guilty feelings towards the employer and his colleagues. However, the action of requesting consent carries considerable risks to the research project. One risk is affecting the impartiality of the research results. "Case studies" in some commercial websites demonstrate the impact of partiality on quality of information. Typically such cases are not detailed and are void of any critical content. Another risk is a permanent delay in getting the research to a conclusion. Commercial organisations, especially large organisations, are simply not set up to deal with such requests swiftly. The problem is made worse by the sheer number of parties involved in a typical ISD project. To be fair to all parties, if one party is asked for consent, all parties must also be asked. The risk is significant for one or more of the companies *not being able* to yield consent, rather than intending to refuse.

The alternative of keeping the observation covert is not attractive. This is because there is an implicit deception on the part of the researcher to the organisations and individuals involved in the projects. The issue of covert research has been much debated (Dench et al., 2003). However, SRA (2002) states in its guidelines:

> "... it would be as unrealistic to outlaw deception in social enquiry as it would be to outlaw it in social interaction. Minor deception is employed in many forms of human contact (tact, flattery etc.) and social researchers are no less likely than the rest of the population to be guilty of such practices. It remains the duty of social researchers and their collaborators, however, not to pursue methods of enquiry that are likely to infringe human values and sensibilities."

While discussing confidentiality and anonymity, Saunders et al. (2000, p. 137) suggest that "covert study" as "a deceitful yet benign way".

The author has reached a conscious decision that apart from the general support from the employer for the author to study the subject, no further specific consent is sought from the clients, the suppliers and the individuals involved in the cases. To a large extent, this decision

is based on the author's firm belief that all parties in the cases studied, including the author, have a unified goal in turning all the projects into successes. This is different from some other social inquiries with aims to reveal social activities involving immorality and illegality, where researchers may have diverse goals from those being studied. In the meantime, the author is clearly aware of the need to respect "human values and sensibilities". This is achieved by observing strictly the following ethical principles (SRA, 2002; Trochim, 2002):

- Anonymisation: all cases are documented carefully to avoid the companies and individuals to be identified. After all, the purpose of the study is not about any organisation's or individual's success or failure in the projects. Rather the research focuses on the process and the economic forces that each party represents;
- Confidentiality of records: While anonymisation disguises the identities, the security of the records prevents uncontrolled access. While some risk of disclosure is always present, the author endeavours to take reasonable steps to restrict access to the original materials and the compiled case studies;
- No harm to the individuals and companies: in addition to the above steps, extra care is taken in the writing of the case studies to respect the individual's dignity and not to cause any harm to the individuals and the companies, so far as there is no sacrifice of impartiality. This is in acknowledgement of the fact that, however well the anonymisation is carried out, the participants will have little difficulty in identifying themselves in the roles and responsibilities described. The subject of the study is not any individual as such, rather it is the roles they play within the cases.

In summary, the research has been initiated with significant potential benefits to organisations and their members. To achieve the research objectives, a difficult decision has been made to document cases only with a general agreement from the employer to support the research. Such a decision has not been made lightly and is supplemented with the ethical principles of anonymisation , confidentiality and no harm to the participating parties.

2.8 Summary

The study adopts the pragmatic mixed-method approach in an attempt to find the root causes of ISD project challenges and possible remedial solutions. The research strategy is exploratory and theory-building rather than theory-testing. The main research evidence comes from three participant observation-based case studies with complete membership roles. The first-hand cases are supplemented by published surveys and case reports. The IBE reasoning strategy is adopted to guide the process from observations to understandings with the help of contrastive and analogical analyses.

The author recognises the limitations of the overall research design and the techniques used for data collection and data analysis. First of all, the lack of direct empirical testing of the proposed hypothesis or explanation is no doubt a shortcoming. The participant observation method for acquiring first-hand research evidence carries with it an inherent weakness of subjectivity, which is limited to a certain degree by cross-checking with available project documentation. Commercial sensitivity also features highly in the author's mind throughout the research. As part of the ethical consideration, the author adheres to the confidentiality of records and anonymisation of identities. The IBE reasoning strategy, epistemologically sound as it may be, still appeals to intuition for the degree of "goodness" for the resultant explanations. While bearing these methodological limitations in mind, The author believes that research presented in this thesis will stimulate debates in the field of ISD project management research.

Chapter 3: The "Software Crisis" and the Current Solutions

"The development of large applications is one of the most hazardous and risky business undertakings of the modern world,"
(Capers Jones, quoted in Vowler, 1999)

3.1 Introduction

In the early days of computing, hardware and software were tightly coupled. Software applications were developed for specific hardware and maintained accordingly (Friedman, 1989). From the 1960s onwards, software became more separated from hardware. General purpose programming languages like FORTRAN and COBOL enabled software to be developed independent from hardware. From the 1970s onwards, with the availability of cross-platform operating systems like UNIX, general-purpose software systems have become widespread. However, there have been many troubled and failed ISD projects in the process of developing software systems. Though the majority have not been subject to detailed scrutiny (May, 1998; DeMarco and Lister, 1999; Yourdon, 2002), sufficient evidence has been accumulated for the state of software development to be referred to as the "software crisis".

This chapter first of all considers the definition of an information system by each system element (Section 3.2). Section 3.3 examines critically the success and failure criteria for ISD projects. The differentiation of project success and product success is suggested. Sections 3.4 and 3.5 examine some of the evidence of the "software crisis" in published case studies and surveys. The available case studies and surveys provide not only empirical project descriptions and statistics but also some perceived root causes and corrective actions, which must be considered in the search for alternative explanations and solutions. One particular group of solutions includes many information system development methodologies (ISDMs),

which are examined briefly in Section 3.6. The on-going ISD project challenges demonstrate the failure of the existing solutions in resolving the "software crisis". An attempt is made in Section 3.7 to explain the paradox of many solutions and little actual progress in overcoming ISD project challenges. Section 3.8 provides a summary for this Chapter.

3.2 Defining an Information System

This thesis focuses on information system development projects. In order to do so effectively, it is important to be clear exactly what an information system (IS) is. In this Section, a critique is made of some currently available definitions in the literature. A refined definition is suggested for the discussions to follow.

3.2.1 Current IS Definitions

An IS can be defined from various perspectives. A typical definition can be found in Bocij et al. (1999):

> "An information system is a group of interrelated components that work collectively to carry out input, processing, output, storage and control actions in order to convert data into information products that can be used to support forecasting, planning, control, coordination, decision making and operational activities in an organisation." (p. 27)

Bocij et al. supplement the definition by describing IS as relying on "people resources, hardware resources, software resources, communication resources (including networks and the hardware and software needed to support them) and data resources". Though hardware, software and communication resources are given dedicated treatment in the book, no such privilege is given to data resources.

Britton and Doake (1996) provide a definition describing a slightly different set of system "elements":

> "...as comprising the software, documentation, method of operation, hardware, users and operators which together make up the software system." (p. 4)

In this definition, the element of data is missing altogether, and documentation, users and method of operations (business processes) are included as part of an IS. While there is little dispute about including hardware and software, there are clear differences regarding whether the definition of IS should be extended to data, documentation, users and business processes. The following sections look briefly at hardware and software as system elements and examine whether the other items should be regarded as IS elements or not.

3.2.2 Hardware

Bocij et al. (1999) describe hardware as "the physical components of a computer system", including "input devices, memory, central processing unit, output devices and storage devices" (p. 47). In addition, hardware components are also essential for networking and communications. Compared with the software cost, the proportion of IS hardware cost has been dramatically reduced (Boehm, 1976; Vienneau, 1996). Though there are issues in selecting and acquiring hardware in ISD projects (e.g. rapid depreciation, see Newell, 1995 for an example), they are not the focus of this study.

3.2.3 System Software and Application Software

In addition to classifying software products into PPSW, COTS and CBSW (Section 1.2), an alternative is to group them by the purposes they serve. There are two main categories: system software and application software (Bocij et al., 1999). System software manages hardware resources and conceptually sits in between hardware and application software. System software includes operating systems, database packages and network software etc. To client organisations not in the software development industry, system software is usually acquired as PPSW. Application software "enables users to perform specific information processing activities" (Bocij et al., 1999, p. 102). Common examples include word processing packages like Microsoft Word and enterprise resource planning (ERP) applications etc. Application software, also referred to as software applications, may be acquired as PPSW,

customised from COTS or developed as CBSW. The focus of ISD projects is usually to develop CBSW and COTS in such a way that they can be successfully integrated with other system elements in client organisations to become coherent systems in order to meet client organisational needs. The success or failure of application software development, either based on COTS or developed as CBSW, is critical to the success of resultant systems and ISD projects. In fact, the term "software development" is often used to mean designing, developing and delivering information systems (e.g. Pfleeger, 1998). This study focuses on managing such development activities by using external suppliers in ISD projects.

3.2.4 Data as Part of an Information System

Modern business systems have their origin in data processing (Friedman, 1989; Davis, 2000; Ward and Peppard, 2002). The business orientated programming language COBOL was designed to process business data in the 1960s. Though technology has moved on, the fundamental raison d'être for many information systems in organisations has remained the same. The difference is that there are many more varieties and much higher volumes of data to be processed. A system might have reference data and operational data. Though a standard definition does not appear to exist (Howard, 2003), reference data (also called static data) are part of a system's configuration and are essential for it to operate. Operational data serves business needs (see e.g. 4TRESS, 2003). A software application may be able to compile and execute without operational data, yet it will not be able to serve its intended business purposes. Data, like hardware and software application, is an integral part of an information system.

The interactions between data and application software can be complex. Data might travel through a system and be manipulated in many possible ways:

- Raw data input into a system through user interfaces or system interfaces;
- Raw data stored;

- Raw data processed and transformed;
- Transformed data stored;
- Data transferred to another system;
- Data referenced for validation or other manipulations;
- Data retrieved, analysed (aggregated);
- Data deleted;
- Data formatted and printed onto some output (including onto a screen);
- Data logged as exceptions to be processed manually.

Some of the above operations may logically follow one another. Some may be limited to certain data types. A simplified data lifecycle model might be represented in Figure 3.1.

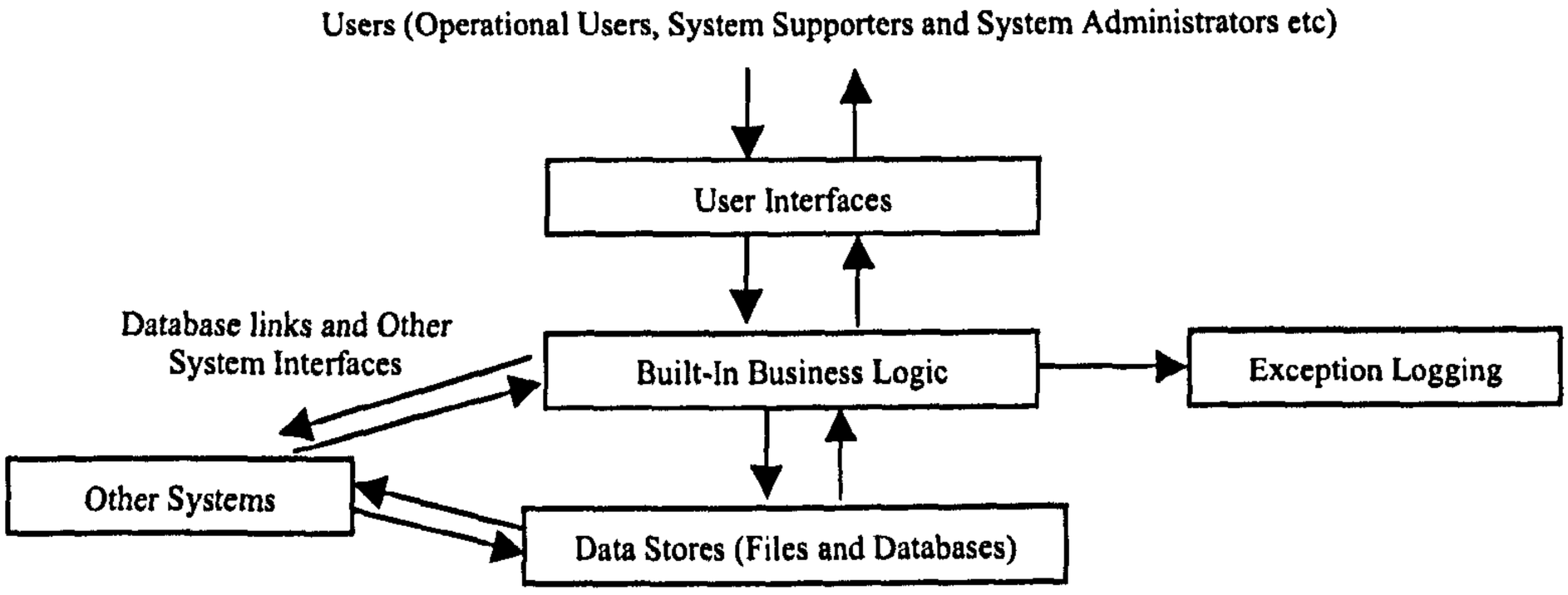


Figure 3.1: The lifecycle of data in an information system

As an example, data might be entered by a user through user interfaces. It enters the system, is stored and then processed by some business logic. The result might be stored as well and also transferred to other systems (inside or outside of an organisation). The fact that a data item has been transferred may also be recorded. Sometimes the source system expects a return message from the target system for confirmation. Stored data might be called up by users through user interfaces. However, user interfaces are often designed for data entry and may only display a limited set of data available in the data stores. When a system encounters difficulties in processing certain data (e.g. no business logic is programmed to handle certain data sensibly from the system's point of view --- dividing by zero being a classic one), it

usually is expected to log the difficulties and inform relevant users. In short, the interactions between application software and data are highly complex and vital for the success of an information system.

3.2.5 Documentation as Part of an Information System

Few people disagree that necessary documentation is a "vital" part of an information system (Brooks, 1995). Depending on the perspective, documentation can be categorised in a number of ways. Pfleeger (1998, p. 267) refers to all "descriptive material" embedded in software source codes as "internal documentation" and others as "external documentation". Within an ISD project, documents might be grouped into project working documents and system documents according to their purposes. Project working documents are produced to assist the running of the project. For example, project schedules, budget projections etc. System documents are written to describe the target system to be developed. They may include architecture descriptions, software system specifications, installation guides, lists of known defects and issues, user guides, and system support guides etc. System documents relating specifically to the application software may also be called software documents. When software application development is outsourced, it is critical for suppliers to deliver quality software documents. System documentation must be up to date and contain sufficient details to be of use to users of various kinds.

3.2.6 Business Processes as Part of an Information System?

A business process can be thought of as "a series of steps involving activities, constraints, and resources that produce an intended output of some kind" (Pfleeger, 1998, p. 44). In the business process re-engineering (BPR) literature, a process has been portrayed as "a number of roles collaborating and interacting to achieve a goal" (Ward and Peppard, 2002, p. 217). These are high-level definitions and give the impression that business processes can be clearly defined and enforced. When it comes to implementing business processes through new

information systems, however, this may be problematic. There are a number of possibilities. One is that the new system automates part of existing processes. Such an implementation may involve usual data inputs and outputs, but with a new system imitating the old one, whether the old one is computerised or not. How closely a new system can do that will make a difference in terms of how comfortable and how much extra training a user might need. Importantly, the new system must maintain the business process consistency, including data input format, data transformation, data storage, data transfer format etc. Another possibility is that a new system is implemented as part of business process change programme, whether intended or forced upon. This may be risky if the new process has different business logic and requires different supporting datasets. There are a number of things to consider. First of all, how can the legacy data be migrated into the new system? This is typically done through a data migration exercise. It is difficult to ensure that legacy data is correctly represented in the new system due to format difference and data interpretation. Secondly, new processes embodied in a new system must connect with other related business processes that are not subject to changes. One of the biggest challenges in acquiring COTS applications is to decide whether the client has to change its processes to suit the applications, or the applications should be customised to suit the client organisation in areas where the built-in processes do not match those in practice. For example, Gage (2002) reports that a human resources management system from PeopleSoft catered for 80% of a client's business processes. The client had to change the other 20% to suit the software system. In other words, business processes may not be defined and enforced at will when they are linked to information systems.

A business process involving an information system usually straddles across users and software applications, often *linked* by data. This can be seen in Figure 3.1 with data flowing from users to various layers of systems and between systems, both internal and external. In addition, to maintain system integrity and to assist system auditing, a significant amount of

data is generated in the process to record transaction details both within and across systems. The meanings of these data items can only be understood through understanding relevant business processes that generate them. Such process knowledge is sometimes found in software and system documentation and sometimes only found in the minds of the expert users. Thus it would be simplistic to list business processes separately as part of an IS. Rather it might be more appropriate to treat the embedded part of business processes as an integral part of an information system (the software application and the corresponding documentation) while treating the remaining part to be outside of the system. Such an approach makes it clear that business processes cannot be delivered in "one fell scoop" with an information system. Instead, they have to be understood by users and absorbed into business practices. It also highlights the need for client organisations to be alert to the impact and potential disruptions of new information systems to business operations (e.g. Gattiker and Goodhue, 2002).

3.2.7 Users as Part of an Information System?

Users often form part of an overall business environment and must be considered throughout the process of designing, developing and implementing information systems. But users cannot be considered as part of an information system within the context of ISD projects. Bocij et al. (1999) might be correct in emphasizing that an information system relies on people resources to be operated. Unlike other integral system elements (e.g. data), users interact with information systems through implicit and explicit processes. If an information system is considered as the output of a project, it is clear that users cannot be considered as part of the project deliverables.

3.2.8 A Recommended Definition of an Information System

Based on the above discussions, a refined definition of an information system, considering the types of software and data, can be recommended as follows:

> An information system is an integrated entity that may serve specific

organisational needs. The elements of an information system may include necessary hardware, system software, shared network facilities, application software, reference data, operational data, necessary interfaces and related documentation.

In the above definition, business processes are not specifically mentioned and users are excluded. Different system elements are integrated into an information system usually according to some defined architecture. A possible schematic architectural representation is shown in Figure 3.2 (partly based on Josittus, 2001 and Shaw & Garlan, 1993). The expected software artefacts from a supplier in an ISD project might be illustrated with the shaded areas. For possible alternative IS architectural representations see Josittus (2001) and Shaw and Garlan (1993), and Pressman and Ince (2000) among others.

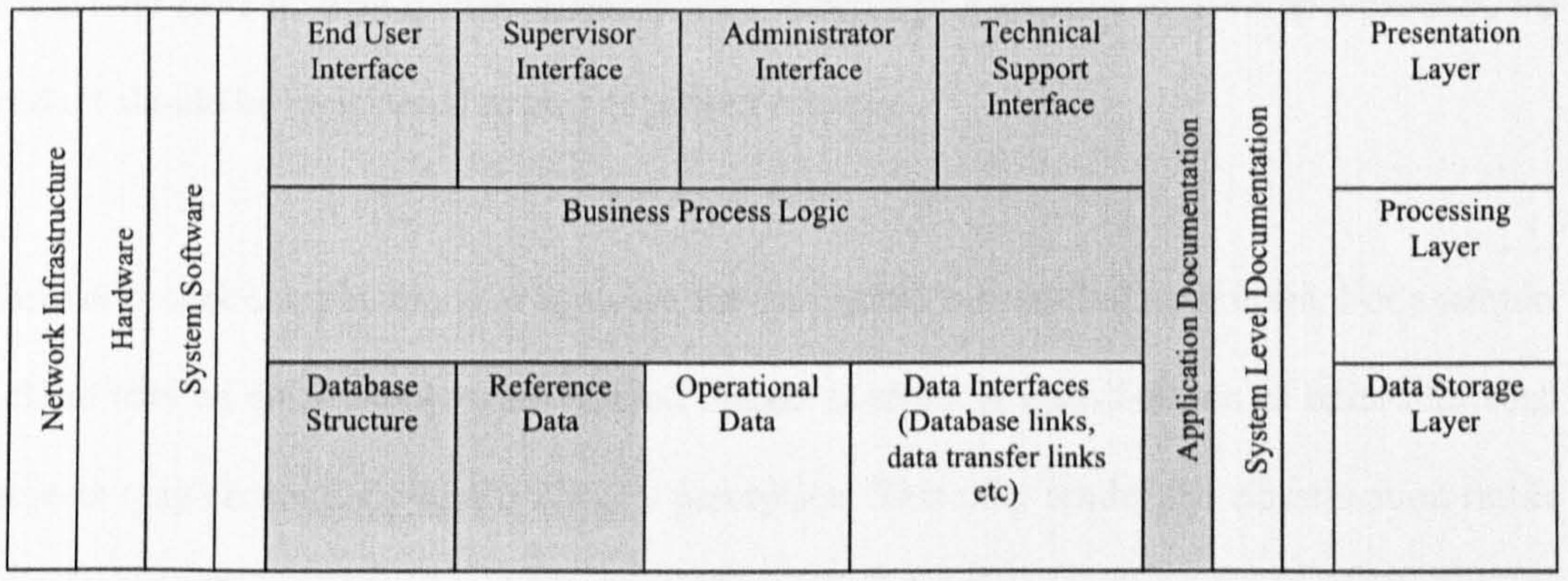


Figure 3.2: A schematic IS architecture

3.3 Success and Failure Criteria of ISD Projects

Having examined the elements of an information system, the following sections discuss the management of ISD projects, starting by examining the success/failure criteria with an emphasis on those found in IS/IT literature. This is followed by examining the "software crisis" with published case studies and surveys found in the literature (Sections 3.4 and 3.5).

3.3.1 The Traditional Approach on Project Success Criteria

The PMBOK Guide (PMI, 2000) does not give a definition of project success/failure criteria. In fact, there appears to be "a lack of agreement concerning the criteria by which success is judged" (Crawford, 2000). The traditional criteria of cost, time and quality, the so-called "Golden Triangle" (Gardiner and Stewart, 2000) or "Triple Constraint" (Rosenau, 1998) have often been used. These criteria are relevant but somewhat incomplete and misplaced. The incompleteness is obvious if we ask "quality of what". Quality does not stand on its own. Quality is a collection of concomitant characteristics of something. That "something" is the end product a project aims to create. To address project quality concerns, the product cannot be overlooked. For a construction project, the product may be a building. For an ISD project, the product is an information system. A product can have basic functions and features, each of which may have its own quality dimension(s). When a project aims to develop a product, the product should be considered as part of project criteria.

There are other complications in applying the traditional success/failure criteria. For example, a client may be quite happy to see a small budget overrun. A classification of failure for such projects may contradict with the client's perception. This may render the classification rather meaningless. To cater for this difficulty, a tolerance level for applying success/failure criteria is sometimes suggested. For example, Capers Jones classified as "best in class" projects those that show "cost over-runs are less than 5% and schedule over-runs are less than 3%" (Vowler, 1999). However, no consensus on the actual level of tolerance is realistic since each client has its own expectation.

Sauer (1993) proposes the concept of "termination failure" or "terminal failure". That is to say, a project fails when it is cancelled. In Sauer's own words:

> "So long as the project organisation can command the resources and power to

> sustain its system, it will not be counted a failure because it is serving some organisational purposes." (p. 30)

Sauer's definition focuses on defining failure rather than success. In this sense, it represents a different approach to the project success/failure criteria. However, it seems to confuse a project with its intended output. A project always has a finite life by definition since it is a "temporary endeavor" (PMI, 2000). Therefore the termination of a project organisation could well be pre-planned and has nothing to do with any project failure. Indeed, a project has to be terminated after it has successfully achieved its objectives. Yet the output of a project, usually a product of some sort, is expected to last and to be used after the project. For example, a building is constructed as a product of a construction project and is expected to remain long after the project finishes. An information system is developed and then operated to improve business efficiency and productivity after the project is closed down. Therefore, the focus should be on the product of a project rather than its organisation.

Standish Group Report (1995) suggests three "resolution types" for ISD projects:

- Resolution Type 1, or project success: The project is completed on-time and on-budget, with all features and functions as initially specified;
- Resolution Type 2, or project challenged: The project is completed and operational but over-budget, over the time estimate, and offers fewer features and functions than originally specified;
- Resolution Type 3, or project impaired: The project is cancelled at some point during the development cycle.

Such a scheme avoids the simplistic dichotomy of success and failure. It is useful in highlighting the scale of the "challenged" projects as well as the "failed" projects. However, this seemingly comprehensive scheme also has its shortcomings. For example, a project may deliver and implement a software product without being cancelled during the development cycle. However, shortly after the project is completed, the client may realise that the deployed

software product does not meet operational needs and is therefore either replaced or scrapped. This is what happened with the Computer Aided Despatch (CAD) System Project undertaken by London Ambulance Service in 1992 (Page, 1993; see also Section 3.4.1 below). With the above classification, it is not clear if this type of project is "impaired" or "challenged".

Maude and Willis (1991) propose a different approach:

> "Software development projects can be said to fail if, for whatever reason, it would have been more economic not to have run the project at all." (p. 15)

This definition relates the perceived value by the client throughout the project to the project cost. Thus, in the beginning of the project, the client would perceive the benefit/cost ratio to be high for the project to be initiated. If at some point in the project, the cost projection should increase to the point that benefit/cost ratio is 1:1, the client would be ambivalent to continuation or cancellation. Should the cost be increased further or the perceived benefit be reduced further, the client would be looking to terminate the project. Unfortunately, it is difficult to assess the exact benefits and sometimes even the exact costs of developing an information system (Bannister et al., 2001). As a result, it is difficult to implement the above definition in practice. This is especially the case when clients have no option other than to deploy a developed system due to reasons strategic, political or regulatory etc.

3.3.2 Project and Product

PMBOK Guide defines a project as "a temporary endeavor undertaken to create a unique product or service" (PMI, 2000, p. 4). This definition therefore excludes those endeavours whose objectives may be purely to gain information or understandings. However, for projects whose objectives are to create products and services, the definition highlights the central role of the end output, be it a product or service. Since services are often called service products, the term "product" seems to be appropriate to refer to both in the following discussions.

For an ISD project, the relevant product is the intended information system, the core part of which is usually an application software. The success or failure of the application software and therefore the information system may be evaluated according to its functions, features and their associated quality dimensions. A software function may be a specific action or a series of actions serving a particular business process. A product with any missing function is an incomplete product. The word "feature" is often used interchangeably with "function" (SPR, 2002). However, features can be construed as more than just the basic functions. Features can be the differentiators between one product and another. While one product may function in the same way as the other, they may have different features. For example, all consumer cars sold in the market can go from A to B (a basic function of a car), some cars, however, may have power steering, spoilers and alloy wheels (features).

A software product assumes quality characteristics through relevant functions and features. For example, a system may be judged to be "secure" if all relevant functions are secure: including network connections, database connections and user interface logins etc. Each function and feature may have its own relevant quality dimensions. Such dimensions may include:

- Performance, or the response time. For any given function, there is an associated sequence of actions. The measurement from the start to the end point can be precise for each time though results may be different each time depending on a variety of factors. Performance is an important area to focus on when it comes to a product quality. Software performance is a major technical challenge and has often contributed to software project failures (Glass, 1998);
- Usability: "a measure of the ease of use of a system, the user friendliness..." (Maude and Willis, 1991);
- Maintainability/Adaptability;
- Security: security features highly among the governmental and corporate concerns

about software quality. It has been claimed that software security problems are directly linked to "flaws of software development" (Festa, 2001);

- Compatibility: software compatibility is important if a software system operates across less than perfectly homogeneous operational environment. This is particularly so with web applications. The end users of web applications are dispersed around the world with unlimited possibilities of access platforms. Therefore compatibility is a compulsory part of web application's quality criteria.

Functions, features and their quality dimensions are closely linked. Product functions are the basis to which features and quality dimensions are attached. A fundamental requirement for a software product is that it has to function according to clients' requirements. In the celebrated words of Weinberg (1971):

> "If a program doesn't have to work, measures of efficiency, of adaptability, or of cost of production have no meaning." (p. 19)

On the other hand, a software product that works functionally may not serve a client's operational needs if it has poor usability and performance.

3.3.3 A Proposed Approach to Project Success/Failure Criteria

With the above discussions, a possible set of project success/failure criteria might be given at two tiers. First of all, a project may be judged to be successful if it is completed on time, within budget and delivers a successful product. Secondly, for a product to be judged as successful, it should meet or exceed the client's expectation in terms of functions, features and quality attributes. In the case of software products, the quality dimensions include performance, usability, maintainability, compatibility and security etc. In this way, product functions and features are rightly emphasised and quality is made more tangible by being related to product.

Though the two-tier approach brings a conceptual clarity, applying it to define project failure carries with it similar problems of tolerance regarding cost and schedule targets. While a project may be deemed to have failed if the intended product is not produced, what if a successful product is produced with cost and time targets exceeded? To cater for such scenarios, it is proposed that a project should be a "qualified success" if it successfully produces a product with budget and deadlines exceeded. Likewise, a project is called a "controlled termination" if it fails to produce a product and is terminated before it is out of control (Table 3.1). Table 3.1 suggests that projects can have different degrees of successes ("Total Success" and "Qualified Success") and different degrees of failures ("Controlled Termination" and Total Failure").

Table 3.1: Project success/failure categories for projects aiming to develop products

	Product Success	**Product Failure**
Within Cost Budget and Time Schedule	Total Success	Controlled Termination
Over Cost Budget or Time Schedule	Qualified Success	Total Failure

The cost and schedule are considered together in Table 3.1. This is because cost and time are closely related, not independent variables as have been portrayed (e.g. Rosenau, 1998, p. 16). Cost increases are often a result of project delays (e.g. KPMG, 1997).

A key question is then how a product should be judged to be a success or a failure. It would be difficult to give a generic answer. A paraphrase of Maude and Willis' (1991) definition may help: a product is a failure if it would have been more economic not to produce it. This is not practical enough for easy implementation, since it often takes time to measure the value of a product (Kharbanda and Pinto, 1996). However, for ISD projects that subcontract software product development to external suppliers, this definition might be adapted as: a product is a failure if it would have been more economic not to accept it, considering all known costs and benefits. Therefore, a product is successful if it is delivered to the project, evaluated and accepted by the client. A product is a failure if it is either not delivered by the supplier as

agreed, or delivered but rejected after evaluation, considering the cost implication of the rejection. Here the concept of product success and failure is a relative one. For a COTS product, it may be accepted by some clients and rejected by others. For a CBSW product, it is only a success to the client if it is accepted and deployed by the ISD project, given the choice. This concept of product success or failure depends on an evaluation. The result of the evaluation is binary: accept or reject. As a result, the status of a product relative to a particular project is also binary: success if accepted, failure if rejected. The assumption is that the client is rational and capable to make such an evaluation and the subsequent accept/reject decision. The suggested two-tier success/failure criteria are not easily applicable to historical projects without an appropriate phase of product evaluation. However, the proposed concept is used in the design of a new management framework later in the thesis (Section 10.3.3).

3.4 The "Software Crisis": Published Cases

Though there is a lack of consensus on success and failure criteria, there is a general consensus on the lack of overall success in ISD projects. The term "software crisis" has been popular since its first use in the late 1960s (Naur and Randell, 1968; Friedman, 1989). In this Section and the next, the scale of the "crisis" is examined based on published case studies and industry-wide surveys. This helps to justify the need for research into the causes of the "software crisis" and possible innovative solutions.

Considering the total number of ISD projects undertaken by organisations, relatively few have been subject to detailed investigation and analysis (Flowers, 1996). Still, there have accumulated a substantial amount of evidence regarding the lack of ISD project success. The evidence ranges from anecdotes to independent audit reports and inquiries. In this section, three published case studies are reviewed to appreciate the varieties of projects that have run into difficulties. Though the problems identified in these case studies are closely linked with their organisational contexts, they are also representative of problems encountered in general.

3.4.1 The Computer Aided Despatch (CAD) System Project

The Computer Aided Despatch (CAD) system project was undertaken by the London Ambulance Service (LAS) in the early 1990s in an attempt to enhance the ambulance services in London. The system was implemented in October 1992 and was abandoned within two weeks. It was well recorded thanks to an inquiry immediately after the suspension of CAD system operation in November 1992 (Page at el., 1993). The Inquiry Report addresses both the CAD system project management and the operations of LAS as an organisation. It draws a number of diagnostic conclusions and provides corresponding recommendations. The case study is also found in Flowers (1996) and Collins (1998). This case summary is based on Page et al. (1993). All page numbers refer to Page et al. unless otherwise stated.

Though LAS was a complex organisation with a history of an adversarial management-labour relationship, Page et al. find that "there was unanimous support for technology to be used to enhance the delivery of ambulance services to people in London" (p. 45). In addition, "there is very little wrong with the overall concepts embodied" (p. 47) in the system. The problem is that the system "for a great many reasons, did not achieve what it set out to achieve" (p. 47). The Report claims that "The biggest question mark must be over the application software" (p. 46), recognising the key role of the application software in an information system.

Though the Inquiry Report was thorough in its coverage of the project and its organisational context, its recommendations appear to have addressed only the immediate causes for the symptoms rather than the "real" root causes. Consider the supplier selection process, for example, the lowest bid supplier was used for the software development. Being a public organisation, LAS was obliged to comply with strict procurement guidelines, in this case, the relevant Regional Health Authority (RHA) Standing Financial Instructions. While pointing out that the guidelines had been followed by the project, the report diagnoses one of problems

to be the supplier selection:

> "These standing instructions provide little qualitative guidance to procurement teams. The emphasis is very much on obtaining the best price. This may be appropriate when dealing with procurement of tangible products where the specifications may be similar, but it is arguably less appropriate when dealing with major IT procurements where it is often more difficult to make quality judgements. We recommend that the standing financial instructions should be extended to provide more qualitative guidance for future major IT procurements." (p. 16)

The suggested action to "provide more qualitative guidance" is to address a perceived cause for selecting a supplier who failed to deliver. The report does not recognise that, in this project, like so many other ones, the project team was selecting a supplier based on suppliers' *promises*, not finished products ready for use. However extensive the qualitative guidance is, it is difficult to foretell a finished product from the initial promises, especially when they are separated by a highly uncertain software development process (see Section 8.7).

Nevertheless, the report is excellent in recording many strands in the unfortunate tale of the CAD project failure. It recognises that some of the key problems are in the management of the project, rather than the technology itself. It dismisses the allegation that the system failure was due to "wilful misuse of the system" by ambulance crews. It even claims that "the system did what it had been designed to do". Instead, the report concludes that "much of the design had fatal flaws that would, and did, cumulatively lead to all of the symptoms of system failures" (p. 41). Though the report does not give specific recommendations regarding the management of design risks, it helps this thesis to focus on the management of application software suppliers and design uncertainty.

3.4.2 The CONFIRM Project

In contrast to LAS' CAD System Project, the CONFIRM Project took place in the private sector in the US (Oz, 1994; Flowers, 1996). AMR, who had developed the highly successful airline reservation system called SABRE, formed a new division AMRIS and proposed

CONFIRM as a highly integrated Computerised Registration System (CRS) for the travelling, hotel and car-hire booking industries. AMRIS sold the concept to Marriott, Hilton and Budget Rent-A-Car, industry leaders in their respective markets. As a result, the four companies formed a consortium called Intrico in October 1987. After a year of preparation and negotiations, Marriott, Hilton and Budget signed "a partnership agreement" with AMRIS in September 1988. They agreed to design, develop, operate, and maintain the new "state-of-the-art" reservation system worldwide. While AMRIS was the "supplier-partner", the other three companies were the "client-partners". Such a venture would have required the highest level of management commitment from the participating companies. That the venture should be a total failure was certainly unexpected. The project ended with a series of lawsuits.

The CONFIRM project displayed some common themes of troubled ISD projects. For example, there were problems in managing client requirements. When AMRIS presented a "base design" in December 1988, Marriott claimed that it was inadequate. The project had difficulty planning for the development. A plan issued by AMRIS in March 1989 was "unacceptable" to the other partners. As development progressed and deadlines were missed, AMRIS continuously reassured its partners and the partners in turn "trusted" AMRIS. The project cost estimate went up from $55.7million to $72.7million and then to $92million. Yet, AMRIS was only able to deliver a partial system at the agreed deadline of June 1992. The partial system had "major problems". The Intrico consortium was disbanded in July 1992 and a series of lawsuits ensued from September 1992.

One thing that has made this case different from many other ISD projects was the participants' responses to the project failures. The AMRIS senior management blamed the project staff as being dishonest in the first instance. However, later AMRIS blamed its client-partners for assigning personnel who "lacked adequate knowledge of the industry" and for failing to specify exactly what they wanted from the system (McPartlin, 1994). The project

also resulted in legal claims and counter claims. These claims were eventually settled out of court.

In analysing the CONFIRM case, both Oz (1994) and Flowers (1996) refer to potential conflicts of interest in AMRIS assuming a role as a client-partner and a supplier. However, no detail of such an argument is given in these sources. This notion appears to come initially from the testimony of a former employee of Intrico and was not subject to any critique (McPartlin, 1994). Conflicts of interest normally arise due to hidden relations (Bott et al., 2001). For example, if a project manager decides to purchase a large software package from a supplier whose sales director is the project manager's partner, there is a potential conflict of interest. Conflicts of interest are usually overcome by disclosing the hidden relations (Bott et al., 2001). The fact that AMRIS was both a partner and a supplier in the CONFIRM venture was an intended arrangement. One may even argue that being both a client and a developer could align the interest of AMRIS with the other client-partners, since AMRIS' own interests were closely tied with the success of the venture.

Based on a suspicion that "unprofessional behaviour might have contributed to the mishap", Oz (1994) suggests a solution through a professional code of conduct. Oz's approach has some generic merit in that every professional should observe a level of ethical codes, written or otherwise. However, Oz does not consider why software professionals should behave unethically. To adopt a moral approach to solve complex technical and organisational problems is perhaps simplistic. Such a moral approach is also problematic since it is difficult to evaluate its cost and benefits, and it is not easy to implement in real world projects.

The CONFIRM project further illustrates the difficulties in evaluating a supplier's competency by looking at its past achievement. The case is also a painful reminder that "partnership" and trust does not necessarily guarantee harmonious relationship in the pursuit

of technology innovations. The legal claims and counter claims demonstrate the possible complex interactions between clients and software suppliers. The case has also shown that IS researches have been inadequate in uncovering root causes for project failures and have sometimes resulted in recommendations that might be considered too general and impractical. On the positive side, the case provides a good example of how clients may cancel an ISD project without necessarily letting it become a "never-ending" story (Webster, 2000).

3.4.3 The CAPSA Project

The CAPSA Project was undertaken by Cambridge University from mid 1998 and the developed system was to go live in August 2000. The CAPSA system was intended to be an integrated accounting system replacing an existing one for the university with some extra functionality. The project turned into an "unmitigated disaster" and received a thorough inquiry from two academic peers from other universities. The inquiry result was presented in two parts: Part A focuses on the management of the CAPSA project (Finkelstein, 2001) while Part B is about the "management and governance issues" of Cambridge University relating to the project (Shattock, 2001). This case summary is based on Finkelstein (2001). The paragraph numbers (§#.#) refer to those found in Finkelstein (2001) unless otherwise stated.

After a chaotic software supplier selection process, Oracle Financials was selected in preference to SAP. In the subsequent effort to implement the system, many deliverables were dropped (§7.59) in a race to meet the "go-live" deadline of August 1st, 2000. Though the project was not cancelled, the "going-live" was described as an "unmitigated disaster" (§7.67). Six weeks after "going-live", "the system was all but unusable" (§7.76). Even after three months' live operation, the system required "at least 11 people for a further 6 months to deal with the critical system issues and stabilise CAPSA" (§7.80). Such a level of system support requirements was not anticipated. In fact, no system support was ever planned for in the first place (§7.72).

The CAPSA case is cited here for a number of reasons. First of all, accounting is one of the most often computerised functions in organisations (Demarco and Lister, 1999). Cambridge University had had two accounting systems implemented before CAPSA. Therefore, the project was not a foray into the unknown, either for the client organisation or for the IT industry in general. Secondly, the project took a COTS approach, which is a "buy" rather than "build" solution that has been regarded as a possible solution to ISD project challenges (Brooks, 1995; Oberndorf and Carney, 1998). Yet the resultant system was "of low quality, being both unreliable and difficult to use" (§3.2). Thirdly, the system enjoyed support from the user communities. This is evident in the users' response to the user training as Finkelstein records:

> "Early summer was not the best time in the academic year to hold training courses for hard-pressed Departmental administrative staff but there was, nevertheless, a substantial enrolment for the courses, and still some will on the part of attendees to 'make CAPSA work'." (§7.70)

The project did not fail due to any lack of commitment from the CAPSA project team and the software supplier either. The project team members "were working exceptionally long hours" (§7.65) and the supplier Oracle Financials was "devoting substantial effort and showing clear commitment to dealing with the outstanding issues" (§7.69). The IT consultancy firm KPMG was engaged to manage the overall project. KPMG seemed ideal for this role since:

> "KPMG were experienced system implementors with a well established technology partnership with Oracle and considerable exposure in Higher Education. ... Above all KPMG were large and reputable, a 'safe bet' for a troubled project." (§7.55)

Yet at an expense of over £1.6million, KPMG made a series of mistakes (§8.17). Surprisingly, KPMG project management team "formally" handed the project over to the University in mid August 2000 while the system was in an unusable state. KPMG received a damning verdict in Finkelstein's report, charged with paying "only lip service to risk management" (§8.17). Clearly, the management of consultancy suppliers like KPMG poses a serious problem on its own. One additional observation is that two project methodologies

were supposed to be used in CAPSA. One was from Oracle called "Application Implementation Methodology (AIM)", designed to provide "templates for high level planning and management, generic workflow definitions, documentation and other standards, as well as testing frameworks" (§7.40). In conjunction with AIM, PRINCE2 project management methodology (CCTA, 1998) was to be adopted. The project reality, however, showed "little evidence" (§8.13) that PRINCE2 was ever followed. AIM, though resourceful, was not effective (§8.13).

This section has sampled three published ISD project cases to provide a glimpse of the possible project contexts and some of the research and investigation results available. It seems that ISD projects have been failing across organisations in different sectors, sometimes involving previously highly successful software suppliers and well-respected management consultancy firms. Other cases studies support such observations with varying degrees of details. Sauer (1993) records a project undertaken by the Australian Government in the 1970s. The high-profiled Taurus Project in early 1990s has featured in a number of case studies in academic journals and also in the popular press (e.g. Flowers, 1996; Currie, 1997; Collins, 1998; Drummond, 1999). Collections of troubled and failed ISD projects can be found in Flowers (1996), Collins (1998) and Glass (1998, 1999). Collins (1998) further provides a catalogue of cases in one of its appendixes. The National Audit Office (NAO) of the UK Government and the General Audit Office (GAO) have provided authoritative accounts for many ISD projects in the public sector (e.g. NAO, 2000, 2001, 2003; GAO, 1992a, 1992b, 1994, 1997, 2002). Academic journals often feature case studies with various research emphases (e.g. Southon et al., 1997; Mitev, 1996). Trade magazines track many high-profile projects with considerable insights (e.g. Computing, ComputerWeekly, Computerworld, IT Week, CIO, CIOInsight etc). Many published cases are used in supporting the analyses in Chapters 7 and 8 (e.g. The RISP Project and the LAS CAD Project in Section 7.3.1; The NIRS2 Project in Section 8.4 and The Libra Project in Sections 8.4, 8.5.1 and 8.7.3 etc).

Though the reported and researched cases may only represent a small portion of the total ISD projects population, they have formed a considerable body of cases materials. The cases are supplemented by industry-wide surveys and reviews, which are examined in the next section.

3.5 The "Software Crisis": Surveys and Reviews

While in-depth case studies provide rich contexts, industry-wide surveys offer a different perspective in helping us to understand the scale of the "software crisis". Standish Group (1995) publishes a survey entitled "The CHAOS Report". The survey was carried out with a sample of 365 respondents from across public and private organisations, representing 8,380 applications. The survey confirms the common pattern of IT projects to be running late, over-budget and with deficient functionality. Many are eventually cancelled. The survey groups projects into three categories: successful, challenged and cancelled, recognising the lack of clear-cut success/failure criteria (see Section 3.2). The statistics indicate overall 16.2% success rate, and 31.1% cancellation rate, leaving 52.7% in the "challenged" category. The financial implication is significant. According to its extrapolation, in the US alone, of a total $250billion spent in 1995, $81billion worth of IT projects were to be cancelled. The Standish Report provides further details of software project success rates based on company sizes (Table 3.2, based on Standish Group, 1995; see also NAO, 2000).

Table 3.2: Software projects success rates by company sizes

	Small Companies (%) Annual T/O between $100million and $200million	**Medium Companies (%) Annual T/O between $200million and $500million**	**Large Companies (%) Annual T/O > $500million**
Project successful: Completed on time (less than 20% over), on-budget (less than 20% over), with all features and functions as initially specified	28	16	9
Project challenged: Completed and operational but over budget, or over the time estimate, and offers fewer features and functions than originally specified	50	47	62
Project cancelled: at some point in the development cycle	22	37	29

In addition, the survey asks the respondents for their opinions on the success factors for IT projects. According to the survey:

> "The three major reasons that a project will succeed are user involvement, executive management support, and a clear statement of requirements. There are other success criteria, but with these three elements in place, the chances of success are much greater. Without them, chance of failure increases dramatically."

The survey further establishes a "success potential" chart and benchmarks four IT projects, three historical and one (Banco Itamarati) was still in progress at the time. It suggests that the chart would be "a useful tool for either forecasting the potential success of a project or evaluating project failure". According to the benchmark, the project in Banco Itamarati "started with a high success probability". Unfortunately, no information has been found regarding the outcome of the project in Banco Itamarati to verify the prediction.

Notwithstanding the specific prediction, the findings in Standish Report (1995) have been corroborated by a number of other surveys. Bronzite (2000) produces a "survey of surveys", including the one from the Standish Group[1]. The average success rate obtained from all these surveys is under 15% (Table 3.3, simplified from Bronzite, 2000).

Table 3.3: A survey of surveys on IT Project success rates

Location	Year	Source	No. Samples	Success Rate (%)
UK	1990	Kearney	400	11
UK	1994	Pagoda	100	11
USA	1995	Standish	365	16
UK	1996	OASIG	45	15
Canada	1997	KPMG	176	16

All surveys ask its respondents about the reasons for failures with similar results, varying usually with the sequence of the specific reasons. Common ones are:

- Poor user requirements
- User not involved
- Weak project planning

---

[1] None of the surveys differentiates project successes and product successes as discussed in Section 3.2. Therefore the project success rates should be interpreted in the traditional sense with potential variations from one survey to another.

- Technical incompetence
- Lack of management support
- Weak business case
- Changing requirements
- No teamwork
- Poor accountability

While surveys may be good in establishing the scale of the problems, they are "not particularly good for exploratory or other research which requires large numbers of open-ended questions" (Saunders et al., 2000, p. 279). Part of the problem is that each respondent would have a different concept of "poor user requirements" and "user not involved". In addition, there is a lack of any mechanism in surveys to ask further probing question like "why user requirements are poor" and "how users should be involved". Standish Report (1995) acknowledges that the study was not "in-depth enough to provide a real solution" to the complex problem of IT project failures.

Systematic reviews of the root causes and corrective actions represent another step in the effort to address the "software crisis". For example, Smith (2001) catalogues 40 root causes numbering from RC01 to RC40 (see Appendix B). However, such a catalogue of root causes is likely to be incomplete even though Smith "was unable to find more than 40 generic root causes". For example, in the case of CONFIRM (Section 3.4.2), there was apparently the problem of project staff behaving unethically by concealing problems (Oz, 1994). Smith's list does not feature such an ethical dimension. Some of the causes defined by Smith are wide reaching and therefore likely to overlap significantly with other causes. For example, RC15 ("Poor project planning, management and execution") may arguably cover many activities in a project. Some root causes are defined with terms like "unrealistic" (RC01, RC05 and RC07) that may be too subjective. Still other causes are given with various assumptions that are

hidden yet disputable. For example, RC08 assumes that it is possible for a buyer to define "non-functional" requirements in the first place, which is questionable (see Section 8.3).

Webster (2000) presents a review based on 120 lawsuit cases dealing with "some form of system failures spanning 25 years from 1976 to 2000". The review gives six "consistent, repeating patterns" that are not mutually exclusive. One is called "The Never-Ending Story":

> "The client contracts with the manufacturer to develop and install a system. The project starts. The completion date slips. It keeps slipping. Each time the adjusted delivery date approaches, the project slips yet again. At some point, one of three things happens: the manufacturer/vendor abandons the project; the client cancels the project; or the manufacturer delivers a system that the client terms wholly inadequate and unacceptable. In some cases, the effort has gone on for years, with millions of dollars spent and little to show for it."

In addition to the descriptions of such patterns and their causes, Webster suggests that the general trend of these legal cases is on the increase. Within the sample base, the number of cases increased from five cases per three years in the late 1970s to twenty cases every three years in the later 1990s, though the review does not claim the numbers to be exhaustive. Like surveys, the causes identified by Webster are often only the immediate causes rather than the "real" root causes. As a result, the recommendations given are often no more than "thou shall" type of advice that is often difficult to implement. Consider one recommendation for an example:

> "...you should do a thorough risk assessment of the entire project at the start and take steps to reduce any risks and protect your interests accordingly..."

Such recommendations have been given many times before. One weakness of such measures is that they fail to address the question of why these "thou shalls" have not been adopted in the first place (see Chapter 8 for root cause analysis, and in particular, Section 8.5 on incomplete contracts).

The report by the UK Government (Cabinet Office, 2000) does not escape this pattern either. This report is a comprehensive review of IT project management in the face of continued

project failures in the UK public sector, drawing lessons from UK and abroad. The Report starts by echoing the survey results already mentioned:

> "In the past, Government IT projects have too often missed delivery dates, run over budget or failed to fulfil requirements." (p. 5)

The report makes a series of recommendations in correspondence to "great many reasons why failures occur" (p. 5). The report dedicates a section on the important aspect of "Procurement and Supplier Relationship". It records evidence of "underpriced or unrealistic bids from potential suppliers". However, instead of searching for reasons that might explain such opportunistic behaviours, the report simply asks suppliers to take actions to "produce realistic plans, including financial, technical, personnel…". Further the report notes that suppliers play the common "bait and switch" game (Johnson, 2000a) by:

> "fielding a highly-skilled team of IT practitioners during the tender evaluation process but substituting weaker personnel after the contract had been awarded" (p. 45)

Having recorded the problem, the report fails to make any specific recommendation to deal with it. The report recommends in general "co-operation and an open dialogue between supplier and client". This is unlikely to be effective. If a supplier chooses to behave this way, it would be naïve, to say the least, that it would communicate such an intention openly. Overall, the report prescribes wide-ranging measures in the hope that they would be cost-effective to implement and improve the ISD project success rate without really understanding the root causes of the observed problems. One such measure is the recommended "Gateway Process", featuring peer reviews combined with project approval processes. The Gateway Process has been implemented by the Office of Government Commerce (OGC, 2001). It may be too early to give a final verdict but there are signs that the Gateway Process appears to have been ineffective. In March 2003, it was reported that OGC was advocating another initiative to establish "Centre of Excellence" teams in each government department (Arnott, 2003b). Much research might be carried out to examine the cost effectiveness of the measures adopted by the UK Government since the publication of Cabinet Office (2000).

3.6 Information System Development Methodologies

Many recommendations given in case studies, surveys and reviews alongside immediate "root causes" have often been loosely called "Best Practices" (e.g. Perks, 2003). To address the perceived problems more systematically, researchers and practitioners have proposed many information system development methodologies (ISDMs). In this section, the concept of ISDMs is explored followed by a brief review of the available ISDMs and their effectiveness in dealing with the "software crisis".

3.6.1 What is an ISDM?

Before discussing ISDMs, it is necessary to comment on the difference between "method" and "methodology". Some authors have been known to equate methodology with "method" and regard the use of "methodology" as a corruption of what a methodology should be, which is the study of methods (Stamper, quoted in Jayaratna, 1994, p. 35). However, if the actual uses of "method" and "methodology" are compared, they may be clearly differentiated. A method is a way to achieve a goal. That "goal" can often be defined, for example, a method to extract iron from the ore with charcoal, or a method to solve a quadratic equation. A user can learn a method without necessary knowing why the method works. A methodology can have a much broader connotation. A methodology does contain the necessary activities and techniques to achieve a goal, but more importantly, it includes an element of philosophy. Jayaratna (1994) emphasises the "why" element in its definition of a methodology as:

> "an explicit way of structuring one's thinking and actions. Methodologies contain models and reflect particular perspectives of reality based on a set of philosophical paradigms. A methodology should tell you 'what' steps to take and 'how' to perform those steps but most importantly the reasons 'why' those steps should be taken, in that particular order." (p. 37)

The "why" element may be entirely missing in a method. Avison and Gitzgerald (1988) likewise thinks that a methodology "is usually based on some *philosophical* view, otherwise it is merely a method" (p. 4, emphasis in the original). In addition, a methodology may serve to achieve a more substantial and less well-defined goal. To do that, there may be a whole series

of concepts and techniques required, not just one. Such a concept of methodology is applicable to undertaking research projects as well as developing information systems.

Based on the above discussions on methodology, an ISDM might be defined as "a recommended collection of philosophies, phases, procedures, rules, techniques, tools, documentation, management and training for developers of information systems" (Maddison, 1983, quoted in Avison and Fitzgerald, 1988, p. 263). This definition is adopted in this thesis.

3.6.2 The Many ISDMs

Apparently there were over 300 ISDMs by 1985 (quoted in Avison and Fitzgerald, 1988). By 1994, the count had increased to over 1000 (Jayaratna, 1994). The total number has since been estimated to be perhaps less than a hundred (Avison and Fitzgerald, 2003). Avison and Fitzgerald (1988, 1995 and 2003) provide an excellent survey of many ISDMs. Jayaratna (1994) offers a thought-provoking conceptual framework for evaluating them. In this section, the historical development of ISDMs is traced.

In the early days of computing (1950s), there was no formalised methodology for software development (Avison and Fitzgerald, 1988). Software programs were mainly either for scientific calculations or for basic data processing needs in commercial organisations. By the end of the 1950s and early 1960s, general purpose programming languages like Fortran and COBOL came into existence. The intended systems became more complex. This is when "software was beginning to overtake hardware as the limiting factor in computerisation growth" (Friedman, 1989, p. 93). Though there were clear signs of software problems in the early 1960s, these problems were regarded as "a new technical challenge that would soon be met" (Friedman, 1989, p. 101). It was not until the landmark Conference on Software Engineering in Garmisch, October 1968 when there was the first open admission of the "software crisis" (Naur and Randell, 1969; Friedman, 1989). During that conference, the

symptoms of ISD projects were reported:

> "Many conference participants reported experiences of large projects which had been disappointing in three main ways. The projects ran on long after their estimated deadlines. They cost more than their budgets allowed. They resulted in systems that did not match performance expectations anticipated in the beginning of the projects..." (Friedman, 1989, p. 102)

The causes identified then included the increasing size of the attempted software systems, the difficulty to measure progress and managing large numbers of people. In addition, the lack of documentation, the lack of design and testing and the lack of skilled manpower were all cited as causes of the "software crisis" (Friedman, 1989, pp. 104-105). Among the suggested solutions, some kind of a "right" way to develop systems was debated. One key recommendation from the Garmisch Conference was to adopt a software engineering approach as follows:

> "First, standardization of program structures, particularly via modularisation, was recommended. Second, there should be a specific ordering of stages in the systems development process. Third, design and programming procedures should be standardized." (Friedman, 1989, p. 106).

This staged approach to systems development was later known as the "Waterfall Model" (Royce, 1970). However, some reacted to such an approach critically. Several conference participants stressed that one could not assume that a system could be designed and specified adequately and completely before coding and testing. Potential problems with the Waterfall Model was recognised by Royce (1970) with proposed actions of iterations and prototyping. Many ISDMs have since been proposed. Iivari et al. (2001) suggest a scheme in order to categorise the many ISDMs available. Based on these researches, current ISDMs may be grouped under five main approaches (Table 3.4).

Table 3.4: Classifications of ISDMs based on their approaches

Approaches	Examples
Structured Approach	Structured Analysis and Systems Specification - SASS (DeMarco, 1979), STRADIS (Gane and Sarson, 1979)
Information Modelling	Information Engineering, Data Analysis (Howe, 1983)
Sociotechnical approach	ETHICS (Mumford, 1983a, 1983b), prototyping (e.g. Lantz, 1987)
Object-oriented approach	Object Oriented Analysis and Design (e.g. Pomberger and Blaschek, 1996)
Systems Approach	Soft Systems Methodology (Checkland, 1981)

There are other ISDMs not fitting into any of the above categories. In particular, a group of "agile methods" based on an iterative approach have recently been keenly advocated, including SCRUM (Schwaber, 1996), the Rapid Application Development (RAD) methodology (Stapleton, 1997), Rational Unified Process (Royce, 1998; Kruchten, 2000), eXtremeProgramming (XP), (Beck, 2000) and various "adaptive" methodologies (Highsmith, 2000; Abrahamsson et al., 2002). The many ISDMs have posed as a methodology "jungle" (Avison and Fitzgerald, 1988; Iivari et al., 2001). A research topic has been to understand, classify and evaluate the available ISDMs (Jayaratna, 1994; Jayaratna and Holt, 1996; Iivari et al., 2001). While the iterative approach will be examined critically later (Section 8.10), it is outside of the scope of this thesis to evaluate specific ISDMs. However, the overall effectiveness of the ISDMs together with the other available solutions is examined in the next section.

3.7 The Paradox of Many Solutions and Little Progress

As reviewed above, there have been many solutions proposed to overcome the "software crisis" since the late 1960s. They range from best practices to systematic ISDMs. How effective have they been? There have been technical innovations in areas like programming languages and techniques. However, there is almost a consensus among researchers and practitioners that there has been no significant progress in overcoming the "software crisis" in general (Abdel-Hamid and Madnick, 1991; DeMarco and Lister, 1999: Eischen, 2002). Brooks (1995) asks with a sense of frustration about his book first published in 1975:

> "How can a book written 20 years ago about a software-building experience 30 years ago still be relevant, much less useful?" (p. 254)

One explanation Brooks offers was that the "software development discipline has not advanced normally or properly" (p. 254). Collins (1998) states on the "Author's Note" page that his book "has not had the slightest beneficial effect". Collins continues: "The numbers of disasters, if anything, appears to be increasing". Bronzite (2000) remarks that "the picture

appears largely unchanged over time" (p. 47). Jerry Weinberg, a celebrated writer in the field of software engineering for over thirty years, remarked that no concepts or methods introduced and adopted in the last 30 years have changed the face of software engineering dramatically (quoted in Layman, 2001). Glass may be a lone voice in saying "it was simply getting harder to find good failure stories" (Glass, 1998). Glass' perception that "software is the dramatic success story of our age" is not necessary contradictory to the "software crisis" under discussion. The software success may be attributed to PPSW while the "software crisis" has been related to ISD projects (see Section 9.3 for a comparative study between PPSW market and ISD projects).

It is a perplexing situation indeed. It has been claimed that "all" root causes of the ISD project challenges have been catalogued (Smith, 2001). Nash (2000) claims "root causes of IT failures haven't changed a bit over the years" despite many available "methodologies" (Section 3.6). Such a paradox of "many solutions and little progress" deserves some explaining. Among possible explanations, two are considered here. One is that the root causes have been correctly identified and the available solutions have the potential to fix the root causes if adopted and applied. But somehow practitioners and their organisations have consistently refused to do so. The other explanation is that the root causes have not been successfully identified. Consequently the current available solutions do not address them. As a result, the practitioners either find the solutions impossible to adopt or have adopted and applied them without much effect. To paraphrase from Humphrey (1998), in the former case, what has been "preached" is the right message and somehow the message has not been adopted and applied. In the latter case, the message is wrong, and hence it has been judged as inapplicable or has been applied without much effect. These two possible explanations are each examined below.

3.7.1 "Why they do not practise what we preach?" (Humphrey, 1998)

Many have adopted the former view and are asking a similar question as Humphrey is. Nash (2000) claims that organisations do not learn. Such an accusation can trigger off a series of questions. For example, it might be asked:

- What should be learned?
- Who should be learning?
- Why the learning is not taking place? etc.

Humphrey (1998) speculates that education is the key to guide the practitioners to do the "right things". Yet questions like the above have not been specifically raised and answered. Of course, answers might be inferred from Humphrey's almost accusatory phrasing of the question:

- "What we preach" is valid and should be learned;
- "They" should learn and practise --- presumably referring to the practitioners;
- "They" do not learn because they have not been educated.

Similarly, Lyytinen and Robey (1999) conclude that organisations have failed to learn. Lyytinen and Robey emphasise that "much of the information from which ISD organisations might learn is readily available from previous experience with ISD projects". The major reasons for the failure to learn are suggested to be "limits on organisational intelligence, disincentives for learning, organisational design and educational barriers to learning". Lyytinen and Robey further suggest:

> "In the parlance of organisational learning, organisations must develop the capability to modify their *theories in use* about systems development." (emphasis in the original)

Lyytinen and Robey do not consider the possibility that there might be a huge gap between the ISD project case materials and what organisations might need for modifying their "theories in use". Instead, assuming what is "readily available" is readily applicable, Lyytinen and Robey conclude that organisations "accept and expect poor performance"!

There are two problems with this approach of explaining the paradox. The first one is that it assumes "what we preach" or what is currently available contains the right solution. The current solutions in the form of best practices, processes, quality standards and methodologies are indeed widely available in books, training courses, websites, and organisational standards. The message has the potential to reach their targeted audiences. There has been evidence to show that the message has actually reached the targeted audience! Brooks (1995) features an anecdote where Brooks came across a reader with his book. When asked to comment on the book, the reader replied that there was nothing in it that the reader had not known already (p. 254). Smith (2001) receives a similar review for his book (Anonymous, 2001). If we take the responses from these readers at their face value, we might be delighted that the readers have got the message, either from the books they have read or from elsewhere. It is not important where they have acquired the message, it is important that they have heard what is being "preached". There has also been evidence that the message has been received at the organisational level. Organisations involved in troubled projects like CAPSA pledged to the use of formal methodologies like PRINCE2 and AIM (Section 3.4.3), thus signalling their awareness of and the desire for the existing solutions. This is not to suggest that all practitioners and organisations have a perfect understanding of all the solutions available. However, the evidence does suggest that practitioners and organisations have been made aware of them and have attempted to learn and apply them. It is a highly questionable claim that organisations “accept and expect” failures. In the case of the CONFIRM project (Section 3.4.2), the supplier AMRIS was part of the organisation that had built the exceptionally successful SABRE system. It is unlikely that the partners in the CONFIRM project went into the joint venture expecting a failure. The fact that many lawsuits have been filed for failed ISD projects (Webster, 2000) indicates that organisations do not "accept and expect" ISD projects to fail.

The second problem is that the explanation assumes that the practitioners and organisations are somehow "irrational". Any individual or organisation, or more generally any economic agent is rational if it chooses to maximise its own utilities, albeit bounded by the availability of information (Arrow, 1986). A rational economic agent is expected to adopt better ways to improve its own economic welfare, given the choice. In troubled and failed ISD projects, all parties face challenges including overworking, high levels of stress, unpleasant meetings of "blame", potential humiliations and accusations if projects fail to deliver. Given the initial commitment each party gives to undertaking an ISD project, a project failure is not the optimal result. If any party finds a way to improve the project so that its own benefit would increase, the party would be expected to do so. To suggest that the practitioners are aware of better approaches, and are convinced that they will produce better results yet refuse to do so is to suggest that the practitioners are irrational. Such a suggestion is contrary to the basic assumption of economic agent's rationality. There is no evidence to show that organisations and practitioners involved in ISD projects are less rational. Due to these two problems, it does not seem to be a plausible explanation that what has been "preached" is the "right message", and practitioners either have not got the message or got it but refuse to apply it.

3.7.2 Have we got the "message" right?

The more likely explanation of the two is that the current solutions do not work. If "what we preach" is actually the wrong message, it is possible that the practitioners may reject the message and refuse to adopt it, or accept the message, apply it without much effect. There is growing evidence that this might be the case. There has been a recent revolt against ISDMs, not from the practitioners, but from some veteran methodologists! Yourdon (1997) urges practitioners to forget about ISDMs and simply get the job done. Rosenblatt (1997) points out the irony Yourdon's book poses:

> "This book contains advice that is disappointingly mundane, considering the source. Here is the man who virtually invented software development methodologies, now telling us to avoid the corporate "Methodology Police" in

order to get the project done on time."

Faced with the proliferation of ISDMs and the continued lack of success, Russo and Stolterman (2000) challenge the underlying assumptions regarding ISDMs. They first identify five commonly held assumptions and challenge them by suggesting the following opposite (ISM in the quotes stands for "information systems methodology", an alternative term for ISDM):

- "Methodologies are not beneficial for the design process";
- "The practice of system design is basically rational (in specific situations)";
- "People cannot identify and make explicit their understanding of good design practices (due to 'tacit knowledge')";
- "It is not possible to communicate design knowledge to practising designers (most ISMs are either not known or not accepted by the practitioners)";
- "It is not possible to change the way practising designers view the design process (ISMs cannot be "inflicted" upon the practitioners)".

Russo and Stolterman do not advocate the above views as universal truth. Rather, they simply try to explain the failures of ISDMs by challenging their underlying assumptions.

The concept and technique of prototyping may serve as an historical example of how "the preached message" might be wrong due to the underlying assumptions. Prototyping as a technique has been used widely in traditional engineering industries. Though it is a familiar and intuitive concept, there is no generally accepted definition when applied to ISD projects. For example, Lantz (1987) defines it as follows:

> "Software prototyping is an information system development methodology based on building and using a model of a system for designing, implementing, testing and installing the system." (p. 1)

The "model" of a system, apart from the specified purpose, has not been described with any particular features. Pomberger and Blaschek (1996) provide a definition that requires a prototype to be executable:

> "A software prototype is an executable model of the proposed software system. It must be producible with significantly less effort than the planned product. It must be readily modifiable and extensible. The prototype need not have all the features of the target system, yet it must enable the user to test all important features before the actual implementation. Prototyping encompasses all activities necessary for the production of such prototypes." (p. 4)

However, the above definition uses terms like "significantly less effort" and "all important features" that may be widely open to interpretation. Nevertheless, prototyping has been advocated as a methodology for the benefit of software development (Lantz, 1987; Maude and Willis, 1991). Brooks (1975) coins the popular slogan "Build two, throw one away. You will anyway!" to support the concept of prototyping. However, the deceptively simple concept of prototyping has its many problems when applied to ISD projects.

First of all, building a prototype system is not an easy task on its own. Maude and Willis (1991) demonstrate that it could be difficult even to agree the objectives of a prototype between a client and a supplier. Secondly, prototyping is a costly exercise requiring precious resources and time. Yet its benefit is difficult to ascertain since a prototype is constructed for learning purpose by definition and it is made to be thrown away. Without a reliable mechanism to conduct cost-benefit analysis, the client would be excused for not wanting to pay for the extra. The technique of prototyping itself does not provide an answer to such a managerial-level question. Thirdly, a successful prototype does not guarantee a successful end software product (e.g. Adolph, 2000). Curtis et al. (1988) even suggest that "the best prototype was a failed effort", indicating that any prototype that misses the problem areas would not provide the required insight. In addition, unlike a clay model for a car, the nature of software entices the growth of prototype to become a full system without any apparent barrier. The consequence is that too often due to cost and delivery pressure, the prototype gets implemented as the final system, as Maude and Willis (1991) demonstrate in one of the case studies. Instead of getting a better system, the client gets a prototype of the system![2]

---

[2] This phenomenon was first pointed out to me by one of my learned colleagues in SLC through his

Brooks (1995) retracts his popular slogan for prototyping by pointing out that the concept of prototyping is based on the "discredited Waterfall Model" (Wilson, 1998). The example of prototyping demonstrates that an intuitive technique that has been successfully used in other industries for product development, and has been elaborated and keenly advocated may be unworkable for ISD projects! In the case of prototyping, it is fortunate that the original advocators like Brooks have had the opportunity and courage to re-examine the concept and acknowledge its fundamental weaknesses. The author makes an attempt in Chapter 8 (Section 8.10) to evaluate critically the contributions and shortcomings of the iterative approach, on which many popular agile methods are based. The evaluation of specific ISDMs is, however, outside the scope of this thesis (see e.g. Abrahamsson et al., 2002).

3.7.3 Why might we have got the message wrong?

Why, having recognised the "software crisis" for over three decades, might we have got "the message" wrong? This is too big a question to be addressed fully here. As a potential contribution to answering the question, it might be conjectured that the situation is related to the chasm between the IS industry and the IS academia. The chasm has existed in the last two or three decades without much change according to Glass (1997). Humphrey's (1998) phrasing of the question with "we" preaching and "they" practising inadvertently reinforces Glass' observation. The chasm may have led to the lack of rigour of many research efforts by practitioners on the one hand and the lack of relevance of the research by the academia on the other (Benbasat and Zmud, 1999; Davenport and Markus, 1999).

The fact that we may have got the message wrong is not necessary helped by the vast amount of raw evidence we have. The key problem might be the lack of coherent theories to explain

---

many years of practical experience with ISD projects within UKOne. The tendency is also described in Marciniak and Reifer (1990).

the available evidence. In fact, the overall theoretical foundation of project management has been questioned (Koskela and Howell, 2002a; Soderlund, 2004). In other words, the "theories in use" (Lyytinen and Robey, 1999) may not be adequate in the first place. As a result, the prescribed message to modify the "theories in use" would be ineffective to overcome problems that are more deeply rooted. Many solutions, especially the recommended best practices, are prescribed to overcome the apparent symptoms as if they were the root causes. The ineffectiveness of such "obvious" solutions might also be explained by the IBE Theory (Section 2.6). The IBE Theory differentiates two types of explanation: the "likeliest" ones and the "loveliest" ones (Lipton, 1991). The "likeliest" explanations are those that fit well with available evidence. The "loveliest" ones are those that can provide most understandings and thus can provide guidance for further actions. The vast number of root causes identified in the popular books about ISD projects (e.g. Brooks, 1975, 1995; Corder, 1985; Royce, 1998; Bronzite, 2000; Smith, 2001) are suspected to be of the "likeliest" type in that they fit well with immediate evidence. However, they are not "lovely" as they cannot guide us for effective corrective actions. They are not "lovely" because they are not unified by coherent theories. Royce (1998) advocates "top 10 principles" for software development but admits that they have "no scientific basis" (p. 65). Further research may be carried out as a study on the success and failure of the epistemological process hitherto used in the IS researches.

3.8 Summary

This chapter starts by examining the elements in an information system followed by a survey of success/failure criteria used ISD projects. A two-tier approach is then proposed to highlight the importance of product success to the overall project success. The subsequent discussions of the "software crisis" and ISD project challenges take place with the awareness that there has not been general agreement on project success/failure criteria. To appreciate the scale of the alleged "software crisis", three published case studies are briefly reviewed. These case studies provide detailed organisational contexts to facilitate the understanding of ISD project

challenges. The industry-wide surveys are then reviewed to demonstrate that the "software crisis" is not limited to any particular client sector. In addition, the historical perspective is explored to show that the "software crisis" was identified as early as the 1960s.

Though many ISD projects have taken place without having ever been subjected to independent review and assessment, a considerable body of evidence has been accumulated over the years to facilitate the development of many solutions. Some of the solutions may be useful in specific circumstances. Their overall effectiveness, however, has been questionable. Prototyping is used as an example to illustrate how an intuitive and well-proven technique in other industries may not have been effective for ISD projects. This chapter ends with a possible explanation to account for this paradox of "many solutions and little progress".

Chapter 4: Case 1 – The WebShop Project

(April 2000 – March 2001)

4.1 Background

At the time of the WebShop project was initiated, the core business of UKOne[3], a major utilities company in the UK, was to serve its customer base of nearly 5 millions energy customers with electricity and gas. In addition, UKOne had a chain of over 100 retail stores selling domestic appliances and consumer electronics throughout the UK (UKOne, 1)[4]. Some of the retail shops had facilities to enable energy customers to pay their bills and marketed energy related products and services. The idea was that energy customers' attention could be turned into consumer goods sales opportunities. Likewise, customers coming into the shops for consumer goods would be exposed to marketing campaigns for energy sales and services. Thus retail and energy offerings were to compliment each other.

Against a background of energy market deregulation and intensifying competition (Ofgem, undated), decreasing gas and electricity profit margins and ever increasing customer expectations, it was recognised by the UKOne management that on-line services using new web-based technologies were an "essential way forward" (SLC, 1). UKOne corporate Board confirmed an e-commerce vision in a Board meeting on 22/3/2000 (SLC, 1):

> "The vision is to create Retail and Energy e-commerce channels that are complimentary to existing store and service offerings, with a supporting capability that will drive customer reach beyond the base. Closer relationships with customers will be developed, retaining existing customers and gaining new customers."

---

[3] All companies directly involved in the case studies have been given fictitious names in accordance to the consideration on research ethics (Section 2.7). Any similarity of the names to those of real companies is purely co-incidental and is not intended by the author.

[4] References found in the case studies in this and the next two chapters are of two types. The internal references are given in the format of (Company, Number) corresponding to the entries in the sections entitled "Internal References" within each chapter. The external references are given as usual in the format of (Author, Year) corresponding to the entries in References at the end of the thesis.

To realise this vision, a project called "WebShop" was initiated. The project was championed by one of UKOne's subsidiary, SubA, which had been responsible for UKOne's existing energy and retail offerings. In addition to providing the technical infrastructure, the project aimed to develop a web-based system with functions to enable energy customers to submit meter readings, view and pay bills as well as purchasing goods. The WebShop Project was formally launched in mid April 2000. Initially, the core project team concentrated on defining the scope. By the end of June 2000, a number of achievements had been made including the selection of the "technology platform", the identification of skill requirements, the selection of suppliers and the collection of functional requirements through workshops.

This case study focuses on the WebShop Project delivery effort, especially the roles played by the suppliers. The project organisation is discussed first (Section 4.2) followed by the selection and contracting of the main suppliers (Section 4.3). The key project milestones are then outlined, emphasising the changes of deliverables (Section 4.4) followed by discussions on cost and quality (Sections 4.5 and 4.6). Section 4.7 provides reflections on topical issues related to this case.

4.2 Project Organisation and Resources

There were many parties involved in the project. The project organisation is depicted in Figure 4.1. The direct client was SubA, represented by its E-Business Project Director. Within SubA, three related but separate teams were holding stakes in the outcome of WebShop Project. The first was the Marketing Team responsible for customer capture and services. The second was the Retail Team responsible for selling domestic appliances and consumer electronics. The third was the Energy Team responsible for energy sales and services. Each team was represented by one or two Product Manager(s). Their role was to specify business processes, agree functional specifications, carry out acceptance testing and "sign off" the

system, among others (SLC, 1). Product Managers reported directly to SubA's E-Business Project Director, whose role was to "[a]gree the scope, objectives, timescales and budgets for the project" (SLC, 1). The E-Business Project Director was supported by a project sponsor, who was a senior business manager at the director level within SubA, providing a link into both the corporate board and SubA's Project Review Board.

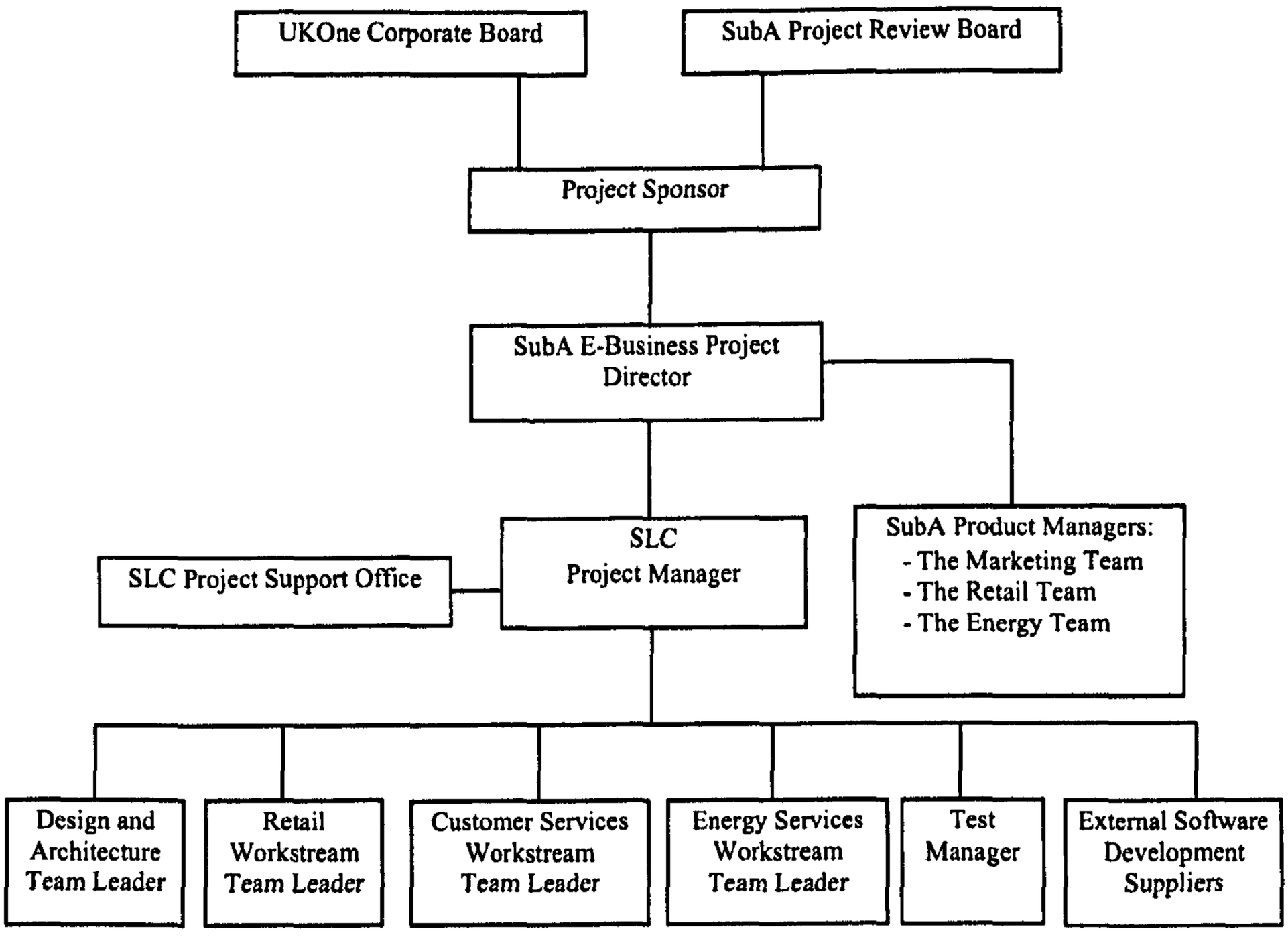


Figure 4.1: The WebShop project organisation

SLC was the prime contractor for the project. SLC provided the core project team members, including the Project Manager, Workstream Team Leaders and the Test Manager. Due to the historical link, SLC had retained a considerable amount of UKOne's business domain knowledge (see Sections 1.4 and 2.5.2 for background information on the link between SLC and UKOne). SLC remained to be the system support provider, with teams of staff familiar with UKOne's systems and skilled at integrating new systems into the existing infrastructure. However SLC lacked a pool of technically skilled and qualified personnel for software development. Therefore, SLC in turn subcontracted software development to various suppliers. The key position of "Design and Architecture Team Leader", otherwise called the

"Lead Architect", was taken up by a consultant from a supplier called MSS. The Lead Architect was to provide the "overall architecture and hardware expertise" (SLC, 24) and was instrumental in technical decisions.

It is worthwhile to point out that the project organisation was dynamic in several ways. First of all, there were changes in the number of suppliers involved. Initially two suppliers were envisaged to carry out the main development. Later, one additional supplier was called in to provide extra development capacity while the need for the services of some legacy systems suppliers for the purpose of system integration disappeared due to reduced scope. Secondly, roles and responsibilities of the key project personnel changed overtime. For example, the Project Manager initially doubled up as the supplier manager (SLC, 1). As the project progressed, the project manager was overloaded and became the bottleneck of project communications. A separate supplier manager had to be added to manage suppliers and reported to the project manager. Another example was the role of the Lead Architect. Half way through the project, the Lead Architect re-defined his role as a consultant providing "advice and guidance" to the project. The Energy Services Team Leader became more involved in the architecture of the solution since The Energy Services workstream was "de-scoped" from the initial phase of delivery (Section 4.4.2).

4.3 Selecting and Contracting Application Software Suppliers

As mentioned earlier, two suppliers were initially envisaged to participate in software development, one for graphic designing, providing the "look & feel" part of the system, and the other for the "backend functions" (i.e. business logic, see Section 3.2.8). NQ and MSS were chosen to fulfil the roles respectively.

4.3.1 Selecting and Contracting NQ

NQ, a company with 250 employees and with offices in London, New York and San

Francisco (NQ, 1), was selected via a process of proposal, review and referencing (NQ, 1; SLC, 2, 3). There was no evidence of competition in the selection process. That does not mean that NQ was necessarily a poor choice. Based on the dazzling presentation (NQ, 1) and well versed proposal (NQ, 2), NQ seemed to be capable and experienced. In the presentation (NQ, 1), there was emphasis on risk management, timeboxing method (see Section 4.7.5), scenario modelling and architecture based system development approach etc. In the proposal document (NQ, 2), solutions were offered with potential issues highlighted, demonstrating remarkable understandings NQ had acquired in the early stage of the project. The selection decision was made soon after NQ's initial presentation. A contract was concluded on 23/06/2000 (SLC, 5).

The contract for NQ's services was between NQ and SLC on a Time and Materials (T&M) [5] basis (NQ, 4). The contract clearly specified NQ's services to comprise of activities covering:

- The design stage
- The build stage
- The system testing stage
- The integration stage
- The roll out stage to deliver software for launch
- The regular steering and team briefing meetings

To carry out these activities, different types of resources were estimated in terms of hours, and each resource type had an hourly rate specified. This gave an estimated cost of £655,776 (excluding VAT) for the services. One important clause in the contract was related to the possibility of exceeding the cost estimate:

> "[NQ] shall not without the Client's prior written consent exceed these estimates. Accordingly, on becoming aware that these estimates will be exceeded [NQ] will notify the Client and, once these estimates have been reached, [NQ] will be entitled to suspend provision of the Services until it receives the Client's written

[5] Time & Materials (T&M) contract is also called cost-plus contract. For a detailed definition, see Bott et al., (2002).

consent to continue." (NQ, 4)

This clause suggests that if the cost estimate is reached without having completed the promised activities, the client is given a choice of accepting what is developed or allowing the development to continue with an increased budget. SubA and SLC actually had to make this difficult choice in the project (Section 4.6.3).

4.3.2 Selecting and Contracting MSS

MSS, a consulting company closely identified with Microsoft and based near the client, was selected to develop the backend functions (the "business logic" or the "processing layer" together with data storage layer in software system architecture terms, see Figure 3.2). The process through which MSS was selected was not well documented. Anecdotal information indicated that it was related to a decision made by UKONE/SubA senior management earlier to establish a "partnership" with Microsoft. MSS was involved in the project as a Microsoft's proxy. Available cost information showed that MSS staff were active in the project as early as April 2000 (SLC, 15, SLC, 16), yet a T&M based agreement with MSS was not finalised until early July (SLC, 13). This looked unusual at least from a process point of view. More unusual was the fact that one MSS member of staff was assigned to the critical role of Lead Architect. This made the technology platform selection a severely biased activity since MSS was closely associated with Microsoft. Any alternative technology platform would have been unthinkable. Nonetheless, an evaluation document was produced in June to assess the pros and cons of Microsoft platform versus that from Sun Microsystems (SLC, 6). The evaluation concluded by recommending Microsoft platform unequivocally. Unfortunately this meant that the project would rely upon a middleware platform called "Commerce Server 2000 (CS2000)" from Microsoft, which was still in its beta version at that time. It meant that NQ would have been unable to perform some of the tasks initially planned for. For this reason, MSS committed to undertake extra tasks to fill the gap while the budget for NQ was reduced accordingly (the budget for NQ was subsequently increased due to the inclusion of a flash page, see Table 4.3

for cost information). Using the project manager's own words, the project was using a "bleeding edge" technology.

4.3.3 Work Allocation Between NQ and MSS

There was no formal Work Breakdown Structure (WBS) documented for the project, but the overall development workload was discussed in a meeting in July among team members from SLC, NQ and MSS. The agreement was documented (NQ, 3), identifying four categories of deliverables:

- Category 1: Static pages (html/JavaScript) – the "look & feel";
- Category 2: Dynamic ASP[6] pages;
- Category 3: ASP pages for help functions;
- Category 4: Components (COM[7]).

These were divided between NQ and MSS, with NQ taking on Category 1 and MSS taking on Categories 3 and 4. Category 2 were split between NQ and MSS (NQ, 3). To integrate the new applications with legacy systems (e.g. retail sales fulfilment database), co-operation from suppliers responsible for legacy systems were also foreseen.

4.3.4 An Additional Supplier IFG

As the project progressed, it became increasingly evident that MSS was not able to deliver what it had promised. In early July the Lead Architect, representing MSS, pledged in writing (MSS, 1) a series of deliverables to be completed by the end of July. This did not happen. The project atmosphere became tense. MSS management commissioned a review of the project by an independent project manager between 07/08/2000 and 11/08/2000 (MSS, 5). A series of recommendations were put forward. One suggestion was to clarify the client requirements by compiling a full list of "feature sets". Each feature would be the basis of man-day estimate

---

[6] ASP stands for "Active Server Page". ASP is a Microsoft technology standard.
[7] COM stands for "Common Object Model", also a Microsoft technology standard.

and project re-planning. This "no-fault" re-planning was completed by 24/08/2000 (MSS, 2). The new plan revealed a need for many additional development man-days. Since MSS did not have the required capacity, an additional supplier IFG, a certified Microsoft partner company based locally and had worked with MSS before, was chosen to supplement the capacity. The selection of IFG was completed within one week or two (SLC, 47, 48, 49) in a desperate hurry. At this point, the project had used up five months time and had only two months to go before the original target delivery date. Yet development in terms of coding had not yet started. Even with the newly mobilised capacity in IFG, to achieve the original delivery date with the original deliverables was impossible. Using the technique of "timeboxing" (Stapleton, 1997; see Section 4.7.4), the scope of the project had to be reduced.

IFG was introduced to the project due to its status as a certified Microsoft partner company and its established working relationship with MSS. From the available documentary evidence, it is not clear if a formal contract was ever signed with IFG. The reports on the number of hours worked, (IFG, 1 and 2) indicating that IFG was paid on a T&M basis.

4.4 Suppliers' Deliverables and Milestones

Before going into the details of the suppliers' deliverables, it is helpful to look at the original intended system architecture (Figure 4.2, simplified from SLC, 6). As Figure 4.2 shows, the proposed three-tier architecture included a presentation layer (the WebServers), a processing layer (the Application Servers and Microsoft Transaction Servers) and a database layer (see Figure 3.2). Such three-tier architecture was adopted to address "issues of reliability, scalability and availability of the overall service without relying on hardware and software" (SLC, 6). However, what was delivered was rather different. To follow the changes of the deliverables, it is helpful to divide the project into three periods according to the changing key milestones. Each of the stages is discussed below.

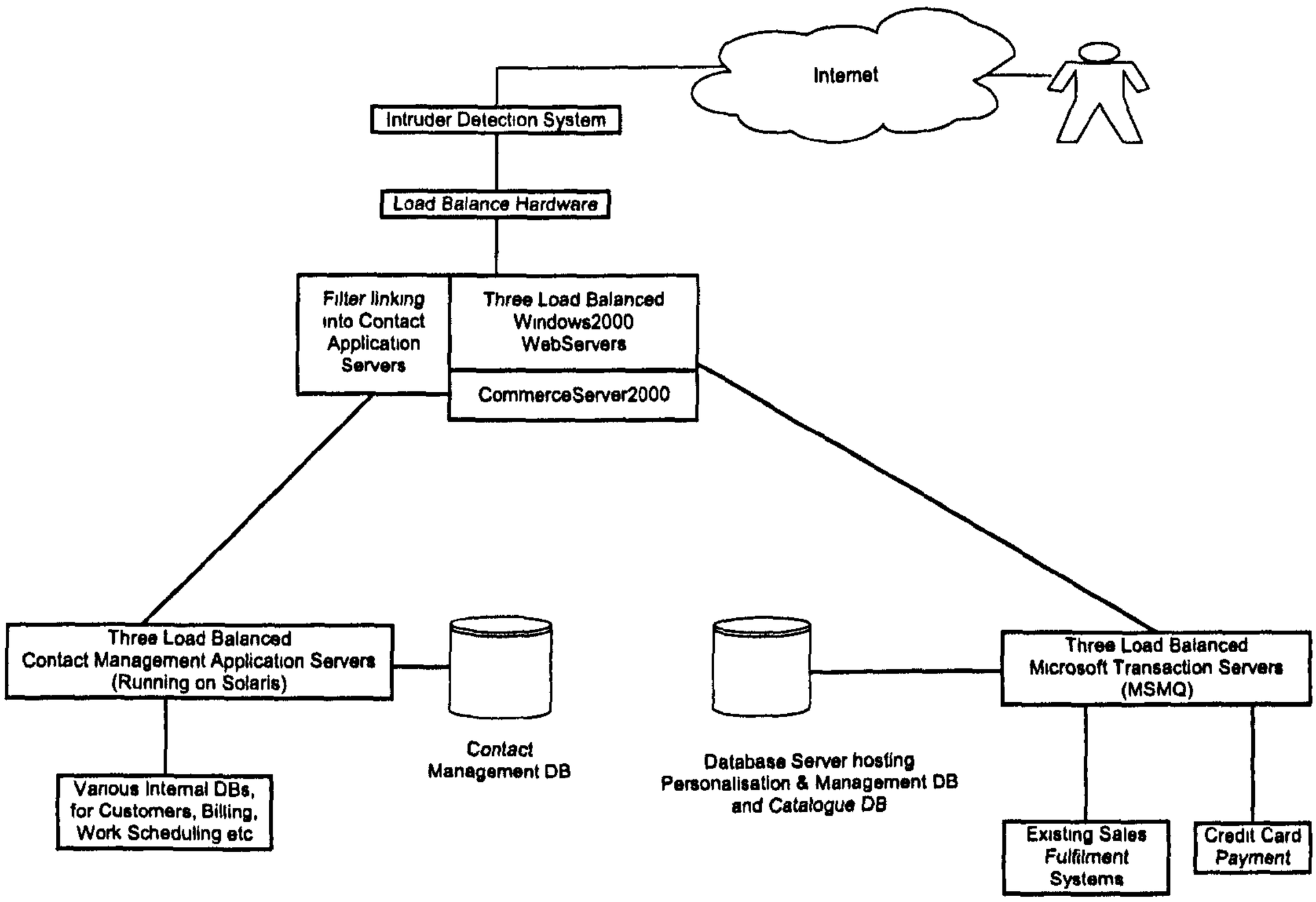


Figure 4.2: The initially intended architecture for the WebShop System

4.4.1 Period 1: From April to mid September 2000 (SLC, 9, 10, 11)

The key milestone was to achieve a comprehensive "soft launch" by the end of October 2000. That included a system with all three parts (Retail Services, Energy Services and the integrated Customer Services) plus the full integration with other systems, some internal, some external. The internal systems for integration included, for example, Retail sales system to check stocks and enable fulfilment of purchases, and Energy Services system to accept customers' meter readings and to provide customers billing information. An example of the external system for integration was a bank payment system. The "soft" in "soft launch" meant the system availability to UKOne's intranet rather than the Internet. By the end of June, it was clear that the project was at least one month behind the schedule (SLC, 12). The project team suggested a "soft launch" with reduced functionality (SLC, 13) as early as July. But MSS' pledge of deliverables in July remained full scale. The "feature-set" approach adopted toward the end of August offered a few different scenarios, some with much reduced functions and

some less so (SLC, 25). Project communication with the client on 05/09/2000 recommended a review to be "undertaken against original scoping documents" (SLC, 35).

4.4.2 Period 2: From Mid September to December 2000

What the project manager presented to the client on 18/09/2000 (SLC, 11) was to separate the system delivery into two releases: Release 1 (Shop Front or Retail Part without Energy Services) and Release 2 (with Energy Services fully developed and integrated). The key milestone was proposed to be the launch of Release 1 on the intranet ("the soft launch"), followed closely by a launch of Release 1 on the Internet by the end of November, while leaving Release 2 to be completed and launched in early January 2001 (SLC, 11). Such a timetable did not meet the client's operational needs. By the end of November, the Christmas peak season would have practically passed from a retailer's point of view, considering the required delivery time for sales through the web channel. As a result, the project team committed to mid November "live" launch for Release 1. As a compromise, the client accepted a further delay of Release 2 to be the end of January (Table 4.1, based on SLC, 27).

Table 4.1: WebShop key milestones as of early October 2000

Milestones	Target Date
Release 1: Shop Front	
Development and Test Complete	30/10/2000
Usability Testing Complete	Mid Nov
Launch	Mid Nov
Release 2: Energy and Integrated Solution	
Development and Complete	30/12/2000
Usability Testing Complete	End of January, 2001
Launch	End of January, 2001

In the process of delivering the Retail part to the Internet, a few requirements were dropped. One was the back end integration with the existing sales fulfilment database. Instead, a manual interface was to be put in place for ordering goods on behalf of the customers, just like the high street shop would do to process a sale. This scope reduction was proposed on 3/10/2000 by the project team and accepted by the client on 10/10/2000 (SLC, 28). The other

major change was to the architecture. The original "three-tier" architecture was reduced to a "two-tier" one with the middle tier removed. This meant that the scalability of the system was compromised. This important change did not go through the established change process (i.e. not found in the project's "Variations Register", SLC, 42). These changes represented major "scope shrinks". Figure 4.3 (mainly based on the author's understanding, with reference to MSS, 3[8]) represents a view of what Release 1 was aiming to be (cf. Figure 4.2).

However reduced the scope was, the critical deliverable was still a developed application from the supplier to SLC so that it could be installed onto an integrated test environment for integration and acceptance testing. Comparing the new plan with the original one, initially the development of the whole system was scheduled to be complete by the middle of September, leaving six weeks for planned testing. The new plan indicated that the partial system was to be developed by the end of October, leaving only two weeks for testing!

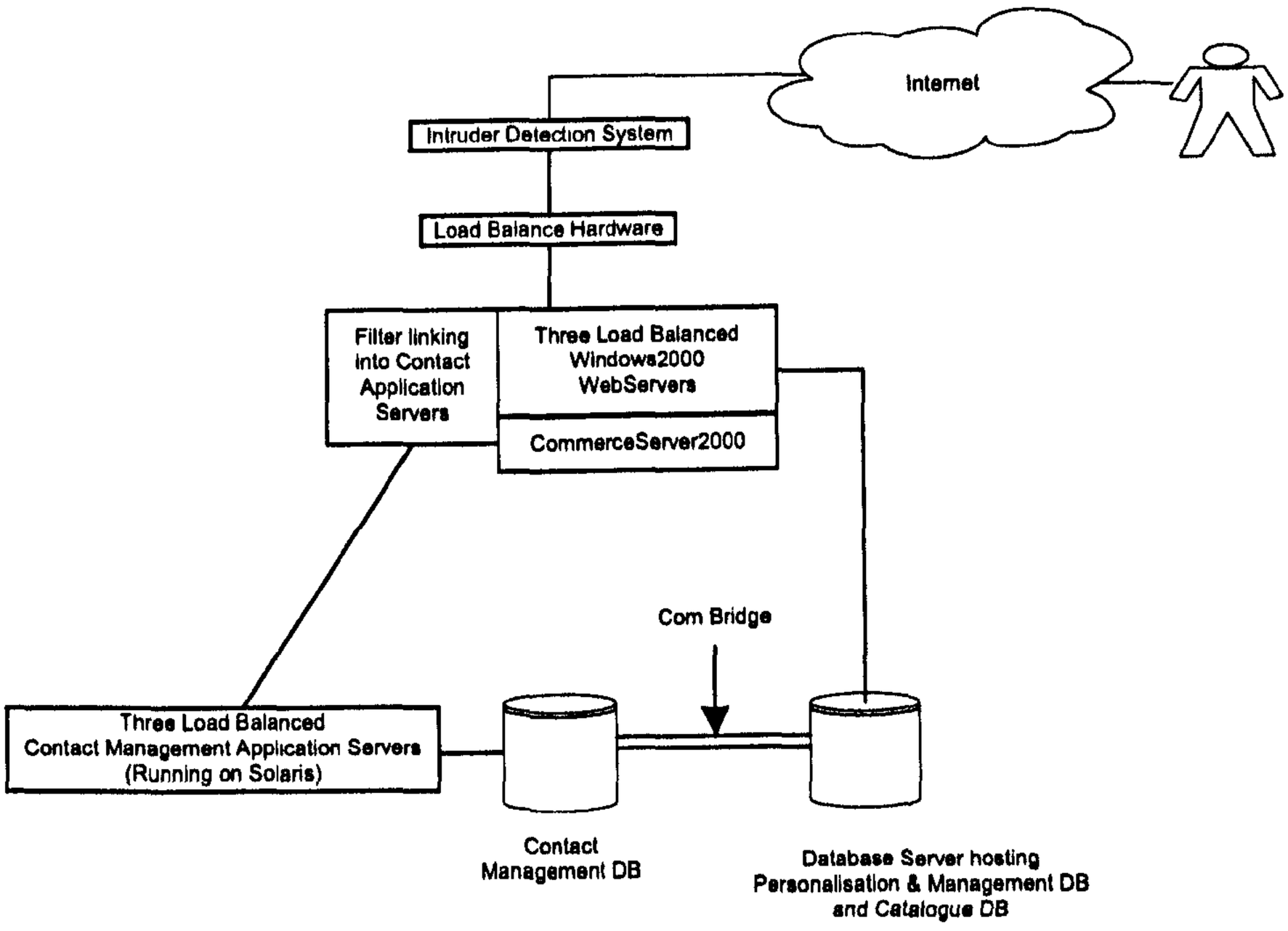


Figure 4.3: Architectural sketch of the delivered WebShop system by February, 2001

[8] The "COM Bridge" in Figure 4.3 was not marked in the initial architecture document (Figure 4.2). However, there was always a need for the two databases to communicate. The missing "COM Bridge" in the initial design document was likely to be an oversight.

SLC received a partial Release 1 on 6/11/2000, and did not receive the rest until the next day, one week later than the revised target of the end of October as shown in Table 4.1. As a result, "Mid November" launch date was stretched and interpreted as 20/11/2000 (one of the few amusing moments in project communications to the client). However, testing revealed many issues and defects. The soft launch date was split further into "soft" soft launch, or Soft Launch 1 (internal intranet based browsing only) and Soft Launch 2 (internal intranet transaction-enabled allowing internal staff to purchase goods, or called Internal Live Launch). The Internal Live Launch became the primary short-term objective at that point, and it was delayed from 20/11/2000 to 29/11/2000 first, then to 4/12/2000 and eventually to 13/12/2000. The delays were due to over 200 defects found during testing which had to be diagnosed and fixed (SLC, 35; MSS, 7).

Juxtaposed with these defects was a delay with Microsoft's production release of Commerce Server 2000 (CS2000). As a result, the Soft Launches were forced to be planned on the basis of the beta version, which had been used for software development but considered unstable for production use. Though CS2000 production version was eventually released on 13/11/2000 (MSS, 4), there was no time to test what was developed on the new version. The initial objective of launching the site to the Internet for the Christmas rush season eventually failed. Effectively, by Christmas of 2000, only UKOne staff was able to purchase goods from the website using an office PC. Looking at the positive side however, a version of the system, however limited, was delivered before the Christmas break, bringing a degree of relief for the project team.

4.4.3 Period 3: From Jan 2001 to March 2001

During this period, the objectives were dominated by transferring the application to the production environment. The plan was to launch the website to the Internet after a two-week stability period and a security audit. The time between soft launch and Internet launch was

typically given as one to two weeks (e.g. SLC, 20). On return from the Christmas break, the project team happily announced that "No outstanding showstopper defects" existed for the Internet launch and it was suggested to launch the website onto the Internet on 15/01/2001 (SLC, 21). However, "Major technical issues" were encountered in moving the application to the production environment (SLC, 22). As an example, one problem was that the database used for holding the sales transaction details was Microsoft SQL Server 7, while working fine in the test environment, was not compatible with the clustering configuration of the production environment. SQL Server 7 had to be upgraded to SQL Server 2000 first. The commissioning of the production environment was delayed a number of times after that (see Table 4.2). The delays had significant impact on the project cost that is discussed next.

Table 4.2: Shifting of the two key milestones of WebShop Project from Jan 2001 to Feb 2001

Milestones	Plan On 9/1/2001 (SLC, 21)	Plan On 23/1/2001 (SLC, 22)	Plan On 30/1/2001 (SLC, 23)	Actual On 26/2/2001 (SLC, 50)
Release 1 Transfer to production environment	15/1/2001	28/1/2001	02/2/2001	01/02/2001
Release 1 Internet launch	31/1/2001	Not specified	16/2/2001	26/2/2001

4.5 Project Budgeting and Cost

The total project cost mainly consisted of provisions for hardware, system software and labour costs for various participants. Hardware and system software costs were budgeted for and spent up front with little controversy throughout the project[9]. Labour costs were mainly for three tasks, namely, the web graphic design and development assigned to NQ, the overall system design and processing layer development by MSS and IFG, and the project management and integration activities by SLC. These costs in terms of the monetary values are presented in Table 4.3 together with their corresponding estimated total cost (Estimated At Completion - EAC) at the time. Also noted in the table is the percentage of each task's cost comparing to the corresponding EAC. It can be seen that the steady increase of the labour

[9] Though there was insufficient provision for testing environment in the beginning of the project, there were three spare servers to use for testing due to the removal of three Microsoft Transactional Servers from the three-tier architecture.

costs of these three tasks both in terms of the absolute monetary values and as percentages of the total project cost (using EAC as the proxies).

Table 4.3: Estimated (Costs) at Completion (EAC) for three tasks from June 2000 to March 2001

Date	NQ		MSS/IFG		SLC		Total Estimated Cost (£000)	Comments (Sources)
	£000	% of total	£000	% of total	£000	% of total		
23/06/2000	655	27	240	10	750	31	2,457	Contract signed with NQ (SLC, 5)
17/07/2000	456	20	300	13	800	35	2,276	Some tasks re-assigned from NQ to MSS (SLC, 29)
05/09/2000	565	24	500	21	785	34	2,327	Flash[10] included in NQ's deliverables; MSS/IFG cost increased due to re-planning. Hardware cost reduced (SLC, 26)
18/09/2000	550	20	799	29	953	34	2,794	Split release strategy requiring extra time (SLC, 11)
02/10/2000	550	20	810	29	963	34	2,815	Launch delayed further (SLC, 18)
17/10/2000	550	22	695	27	882	35	2,549	Retail back-end integration abandoned (SLC, 28, 30)
24/10/2000	550	20	695	25	882	32	2,749	£200k contingency fund added to budget (SLC, 32)
19/12/2000	550	19	873	30	1,025	36	2,870	A new delivery timetable suggested (SLC, 31)
09/01/2001	525	19	877	31	985	35	2,806	Delivery time adjusted (SLC, 21)
23/01/2001	520	18	892	30	1,122	38	2,933	Delivery time adjusted (SLC, 22)
30/01/2001	520	18	851	29	1,122	39	2,892	Minor adjustment on MSS spending (SLC, 23)
13/03/2001	520	18	867	30	1,142	39	2928	Project closedown delayed to 16/3/2001 (v36)

The project started with an initial estimated total budget of £2million (SLC, 7). Suppliers' indicative costs in the beginning suggested that the project was under-budgeted. Throughout the project, much effort was spent on containing cost. First of all, tasks were reassigned from NQ to MSS, hoping to capitalise on MSS' willingness to undercut the competitor's price for the mid-tier development. That hope was dashed when MSS increased their estimate significant as a result of the re-planning (SLC, 35). Due to the nature of the contracts with all suppliers being of the T&M type, the project delays forced the cost up despite the continued

[10] Flash: A technology that enables animation to be displayed on a web browser. For a more detailed definition, see TechTarget (2002b).

scope reductions. Table 4.3 clearly shows the high percentage of the total labour costs. The three main parties' (counting MSS and IFG as one) labour costs added up to 67% of the total budget in the beginning and 86% toward the end. One important point to note is that SLC's cost increased as the project was delayed. There was a significant amount of idle time within SLC since the planned resources were in place yet could not be effectively utilised due to the software products not being available from the development suppliers.

Even the above cost overrun was somewhat under-reported. This was due to some of the costing incurred by SLC team members charged elsewhere under the management instruction. SLC had a system to charge the staff time to different financial codes for different projects and tasks. In that way, project budgets could be tracked and managed. When the WebShop Project's budget was under pressure, some staff were instructed to book some of their time to other codes. Whether the funding was from SLC's internal sources or other UKOne related projects is not the issue. The important thing to note is that SLC's cost figures in Table 4.3 are likely to be under-represented. It would certainly be true to say that no effort was spared to contain the SLC costs within the budgeted limit. Even so, the SLC cost kept increasing well above the original estimate.

4.6 Software Product Quality Control

To ensure the quality of any software product, the client has to make sure that the requirements match the real needs. In addition, the client has to make sure what is delivered match the requirements. The first part is done through requirements management before and during the product development stage, the second through quality control at the product acceptance stage. The essential observation was that neither stage worked well in the WebShop Project.

4.6.1 Requirements Management

Within the WebShop Project, there was no specifically stated and enforced process about requirements management. The activities of generating requirements and having them reviewed and authorised were guided by document templates, whose own quality was variable depending on their sources. The client teams used UKOne's document templates for the functional requirements. Three main functional documents (SubA, 1, 2, 3) were produced at the beginning of the project. They were issued for comments and feedback was incorporated as indicated by a revision history. However, there were a number of problems. First of all, requirements documents contained running commentaries without clearly delineating separate functions. As a result, it was difficult to establish a clear link between a requirement and a specific developed software feature. Secondly, there was no formal process controlling the reviewing, updating and issuing activities. Thirdly, related to the second problem, it was not clear at any moment of time if a document was mature enough to be issued for design and development. A conceptual milestone existed that the requirements should be "signed-off" before development activities could have started. But there were both theoretical and practical reasons preventing the sign-off from taking place. Theoretically the concept of requirements sign-off was considered to be based on the discredited "Waterfall Model" (Wilson, 1998). Practically, initial requirements would have been incomplete and incoherent. Any sign-off would have been taken as the baseline for suppliers' cost commitment. Any subsequent change would have been regarded as variations and might have attracted cost penalty and been blamed for any schedule delay. As a result of these requirements management issues, three main requirement documents were found to be last updated on 27/06/2000, 30/06/2000 and 04/08/2000 respectively in the project archive. Two of the documents were left in the draft status, the status of the other one was not clear. Subsequent changes were not incorporated into the original documents. Though some variations were raised when requirement changes were agreed, the variations did not cross reference to the original requirements documents. The value of the original requirements documents was thus

diminished as the project progressed. Using the requirements as objective quality criteria was therefore not possible.

One variation of the requirements management process in the project was the Rapid Application Development (RAD[11]) approach to develop the "look & feel". RAD methodology dictated that the design should be attempted and refined according to the client's feedback. This was tried with NQ producing the "look & feel" design for the client to review. After "numerous iterations" (SLC, 26), a baseline design was signed off by 18/08/2000 (SLC, 26, 39). On top of that, each web page based on the design was also to be signed off by the client. Although at times, the speed of sign-off was "a source of concern" (SLC, 40), the project manager reported that all graphics and content had been signed off by 12/10/2000 (SLC, 28). A demonstration of the design followed for the first time to a wider audience from the client. It was reported (SLC, 41) that the client had "issues" with the design. An entry in the risk register was more specific, stating that the client had a "dislike" of the design (SLC, 14). By then, it was too late to have any substantial change. The risk was that if "there is a requirement to modify the layout this will affect all pages – and will require substantial rework" (SLC, 14). As a result, no further action was taken to remedy the graphic design. As intuitive as the RAD approach was, the result failed to meet the client's expectation.

4.6.2 Product Acceptance

Another mechanism to control supplier's software product quality is product acceptance. The idea is that a number of acceptance criteria should be agreed between the client and the supplier before software product development. At the product acceptance stage, the software can be accepted if all criteria are met (Pfleeger, 1998). This approach was attempted in the WebShop Project. The acceptance criteria were repeatedly mentioned as one of the key

[11] Rapid Application Development (RAD): a software development approach conceived to overcome the traditional methodology following the Waterfall Model (Stapleton, 1997; Murch, 2001; DSDM, 2002).

dependencies in September and October of 2000 (SLC, 11, 26, 27, 28, 32, 34). The compilation of the criteria was helped by an independent consultant joined the project in early October 2000 (SLC, 43). In a meeting agenda drafted by the consultant, 5 major categories of attributes was listed, covering 12 minor categories of quality dimensions (Table 4.4, based on SLC, 43) as a starting point of considering the acceptance criteria.

Table 4.4: Suggested acceptance criteria

Attribute	Quality
AVAILABILITY	Planned operational hours
	Reliability
	Recoverability
	Integrity
	Maintainability
	Auditability
WORKABILITY	Scalabilty
	Responsiveness
ADAPTABILITY	Viewability
	Portability
MANAGABILITY	Supportability
FLEXIBILITY	Updateability

The client took on the suggestion and produced two drafts of criteria (SLC, 44, 45) with additional categories (e.g. security). These documents went into considerable details regarding some of the technical aspects. References were also made to other testing phases and established data standards (SLC, 45). However, the criteria document suffered from a number of shortcomings. First of all, the criteria did not refer to any previous requirements specification. Secondly, when the established data standards were included, no reference was made to any specific documented item or authority. Therefore the acceptance criteria appeared disjointed with the rest of the project and the client company's general IT and data policies. Thirdly, no option was established as for what would happen if one or more of the criteria were not met. A natural default position would be to extend the project until all criteria were met. But such an option, though eventually taken by the project team, meant heavy penalty in terms of budget overruns and a high level of stress on all project team members.

What was surprising was that the acceptance criteria, once compiled, were practically ignored. The decision to launch the website onto the Internet in February 2001 was based on a different checklist called "Critical Success Factors" (SLC, 46). The approval mechanism conjured up was that each item on the checklist was to be endorsed by named individuals. Most of the named individuals were part of the project team. There was no specific reference back to the acceptance criteria produced earlier.

4.6.3 Quality of NQ Deliverables

In NQ's proposal document (NQ, 2), a few activities were specifically mentioned in relation to quality assurance. Here is an excerpt regarding "System Testing":

> "System testing follows build and involves the development team testing the 'code complete' software for defects against defined test scripts written following design."

Unfortunately no test script based system testing was documented and communicated by NQ to the project team. NQ delivered html pages but they proved to be not compatible with Netscape browser versions. NQ refused to carry out further work without extra funding when they used up the pre-agreed budget (see Section 4.3.1 about the relevant contract terms). Extra coding had to be added by SLC staff later. Fortunately SLC Team was able to do it technically, and with resources freed from the Energy Services workstream to concentrate on the Retail and Customer Services work.

In addition to the html pages, NQ was also responsible for the "look & feel" design of the web pages. As discussed in the section on requirements management, although the design was reviewed to be "excellent" by "all parties" (SLC, 26) in the beginning, it was disliked by the client (SLC, 14). There was no evidence that this issue was tackled.

4.6.4 Quality of MSS/IFG Deliverables

Before IFG was introduced into the project toward in August 2000, MSS was solely responsible for the system design with deliverables including the system architecture. MSS had great difficulties in meeting its own promises to deliver. So far as the architecture was concerned, MSS did eventually issue a document entitled "High-Level Architecture & Design Document" (MSS, 3) on 01/09/2000 under much pressure from the project team. But the document was a draft version and it was incomplete. The document was not known to be ever updated thereafter.

After IFG joined the project, MSS limited itself to developing some core software artefacts and assumed mainly a consulting role for giving advice and carrying out quality assurance work on what was produced by IFG. This was much welcomed by SLC since the project test strategy did call for independent quality assurance team separate from the development team (SLC, 33). However, highly competent (and highly paid) independent quality assurance technical staff were not able to prevent defects and issues. A total of over 200 issues were discovered and logged within a week of IFG's release on 7/11/2000. Table 4.5 (based on SLC, 35) shows the defects resolution progress between 14/11/2000 and 24/11/2000.

Table 4.5: Defects/issues statistics between 14/11/2000 and 24/11/2000

Severity Level[12]	Open as of 14/11/2000	Open as of 24/11/2000
Severity Level 1	44	0
Severity Level 2	99	77
Severity Level 3	74	19
Non Priority	N/A	13

More defects and issues were discovered as the project moved on. Many issues arose from system level testing after the software artefacts were delivered (e.g. browser compatibility testing), indicating a lack of testing by the development suppliers in the first place. Other issues came up due to interactions of different artefacts and platforms.

---

[12] Severity Level of a defect is normally judged by the defect's impact on the system's behaviour. Defects of Severity Level 1 have more negative impact on the system than those of Severity Level 2 and so on.

4.7 Project Issues and Within-Case Analysis

The WebShop Project appears to be a typical "challenged project" in the scheme of Standish Group's project classifications (Standish Group, 1995; see Section 3.3.1). By the initial deadline of the end of October 2000, the project had not delivered a workable system. By the time the project drew to a close (middle March 2001), the system delivered was partial compared to what was planned. A second phase had to be initiated afterwards. As the project was delayed further, the project cost kept going up. In this section, the issues causing the challenges are highlighted and analysed with supplementary project details where necessary.

4.7.1 Project Organisation

An issue that affected the project operation profoundly was the assignment of a lead architect who happened to be a team member of the key supplier MSS. This assignment appeared to have been made by senior management out of a desire to forge a close alliance with Microsoft. The project manager had no prior influence, neither could he change such an arrangement. The impact was both personal and organisational. On a personal level, the lead architect had no loyalty to the project manager, and conversely, there was little respect from the project manager to the lead architect. There were often stormy arguments between the two in project meetings. On the organisational level, the lead architect was the technical authority. Due to the company association, he was closely identified with Microsoft and its technologies. Whether it was wise to go for the "bleeding edge" technology (Commerce Server 2000 at its beta version), the lead architect would be expected to have a strong influence of the technology decision in June 2000 to favour Microsoft. It was a risky proposition yet the project team was comforted by the thought that some vice president of Microsoft was behind the project. More seriously, since the lead architect was the lead designer of the overall system and also part of the development team in MSS at the same time, the design was changed at will without being fully documented and agreed with SLC project team and the client.

The situation was extremely stressful and was alleviated to a certain degree when IFG was introduced to supplement MSS' development capacity. MSS limited itself to developing some core artefacts and turned itself largely into a consulting role providing project advice on design and quality assurance. These changes had the effect of separating design and development to some extent. Another welcomed change was the introduction of a new account manager within MSS to co-ordinate their resources and activities, and thus reducing the direct exchanges between SLC project manager and the lead architect. Unfortunately, these changes happened at a considerable extra cost to the project, therefore contributing to the project cost overruns (MSS, 8).

4.7.2 Supplier Selection

NQ's selection was completed rather quickly. NQ made a presentation on 01/06/2000 and issued a proposal on 08/06/2000. By 09/06/2000, the decision had already been made to select NQ as the design "partner", "pending successful commercial negotiations" (SLC, 3). There was no evidence that any competition took place in the process. The decision seemed to be made before any substantial technical review was carried out (SLC, 3), and before references were taken and price negotiation was concluded (SLC, 3, 4 and 5). Such a process was unorthodox to say the least. This may be explained by the supplier partnership approach assumed by the project and the client. However, it is an example of early supplier selection commitment, much too early considering the desired product was a design output with a considerable level of uncertainty.

MSS' selection was made even earlier. A weekly project report dated 12/05/2000 to the client revealed a task planned but not completed in that week to "Finalise Commercial agreement with [MSS]" (SLC, 36). There was no documented evidence within the project as for exactly when and how MSS was selected. Anecdotal evidence suggests that it was a decision made

outside of the project by SubA's senior management out of a desire to forge a strategic marketing partnership with Microsoft. MSS was closely identified with Microsoft and was in a position to provide Microsoft technologies and services. There was no objective assessment available to indicate that MSS was suitable for the project. To SubA, MSS was seen to be key to the web marketing strategy. To SLC, there was an expectation to have certain technology skills transfer from MSS to SLC project team members. These organisational objectives were neither good nor bad in themselves. But they were to be achieved by imposing on the WebShop Project a key "partner" for reasons not related to the focused project objectives. This had a series of impacts on the project causing many difficulties. It may serve as an example of external inference to project decision-making.

4.7.3 Partnership Approach

Throughout the project the "supplier partnership" approach was emphasised. Within SLC's project team, all suppliers were to be called "partners". This approach was shared by the suppliers, at least in words. NQ claimed it to be the "ideal partner" in its proposal (NQ, 2). The concept of partnership may explain the supplier selection process to be totally devoid of competitive tendering. Unfortunately, at times of conflict and difficulties, there was little guidance and no effective mechanism to manage the partnership relationship.

A specific example was NQ's refusal to carry out system testing, even though that was specifically promised in the contract as one of the activities to be carried out (NQ, 4; Section 4.3.1). From NQ's point of view, it was logical to conclude that it was a T&M based contract constrained by a budget. When the budget was exhausted but the work had not been completed, it was fully justified for NQ to request for further funding. Otherwise, no further work would be done. Indeed, who could have blamed NQ for refusing to carry out work "free of charge"? The "partnership" approach did not appear to have made any difference.

Another example was that MSS failed to deliver a host of items as promised before the end of July 2000 (MSS, 1). To salvage the project, SLC eventually had to bring in another supplier IFG at a considerable extra expense. When the project was delayed further and the overall project cost was exceeding the original budget, MSS continued to enjoy the same high fees as time went by. The "partnership" approach did not appear to have made any difference here either. The observation from this project is that supplier partnership is a slogan with little substance.

4.7.4 The Timeboxing Method

Timeboxing is a technique advocated in the Rapid Application Development (RAD) methodology (Stapleton, 1997; see also Section 8.10.3). This method was much emphasised at the early stage of the project as a key to controlling the project scope, to achieve on-time delivery within the set budget (SLC, 1). In practice, the technique was used aggressively and resulted in significant reduction in the project scope (Section 4.4). However, it did not prevent the project from being late and over budget. From the experience of this case, two problems are observed with the timeboxing method.

One is that timeboxing does not deal with the complex, interrelated nature of software components. It was possible to leave behind a whole workstream like Energy Services, but it would not have made any sense to segregate customer services from the retail part. Selling goods was not separable from delivering the goods. In other words, the required application could only be divided within considerable constraints. The other problem was highlighted by NQ's refusal to fulfil its obligation to carry out system testing on its deliverables. The lesson was that functions may have been completed in its rudimentary form but testing and defect-fixing may require considerable efforts. But NQ used timeboxing as an excuse for not proceeding any further. In other words, quality was "de-scoped" as well as functions. The net result was a poor bargain for the teams carrying out down stream activities. In WebShop

Project, the integration team had to carry out considerable extra work to make the NQ's html pages to be compatible with different browser versions. The observation is that timeboxing could not secure even a reduced system with essential quality attributes to be delivered on time or within budget.

4.7.5 Requirements Management and Variation Control

The approach to requirements management was not consistent in the project, to say the least. At the early stage (from May to early August 2000), business functional specifications were produced by the three workstreams. Discussions took place and feedback was incorporated in many revisions. But the functional specifications remained as drafts (Section 4.6.1) and therefore could be considered as "incomplete". In the meantime, the RAD methodology (Stapleton, 1997; DSDM, undated-a) was adopted from the start of the project (SLC, 37). It was believed that with RAD, it was not necessary to spell out requirements before starting development. This perhaps explains the fact that no functional specification was "finalised" or "signed-off". However, in the review requested by MSS management by an independent project manager, the incomplete functional requirements were cited as the key to many problems and delays (MSS, 5). Fierce arguments broke out regarding the necessity of detailed documented requirements. In the end, MSS won the debate. But instead of reviewing and updating the functional requirement documents already available, the project team spent two weeks preparing a spreadsheet listing all the "feature sets" (MSS, 6). Shortly before the delivery of the software artefacts, all requirements were further distilled into a single page, listing the functions to be in included or excluded in the first release (SLC, 27, 28, 32, 34). No effort was made to update early requirements specification. Requirements specification seemed to become less comprehensive and highly simplified as the project moved forward, leaving many details to interpretation. Therefore, the requirements remained incomplete even at the end of the project.

Closely associated with incomplete requirements is the concept of variation control. This is supposed to reduce "scope creep" in requirements management (e.g. Aniceto, 2003). Strictly speaking, an initial baseline of requirements should be established first. Development activities should be planned according to the baseline of requirements. The baseline should be subjected to variation control. Development activities can therefore vary accordingly and the project cost and schedule can be managed based on requirements variations. In reality, variations appear to have been used more to reduce the project scope in Case 1 rather than to increase it. In total, 35 variations were documented in the "Variations Register" (SLC, 42). The variations are a mixture of requirements clarification, scope reduction and cost increase requests. The single most significant variation recorded was a requirement reduction, taking the integration with the sales and the banking systems out of the project scope. On the other hand, some of the most significant variations were not documented. One example was the removal of a middle tier of Microsoft Transaction Servers, resulted in a system with much less scalability (Section 4.4). In short, despite the concepts of requirements management and variation controls, control on requirements seemed difficult and ineffective.

4.7.6 Resource Capacity Planning

There appeared to be a real problem with resource capacity planning in the project. Resource capacity planning refers to arranging the right amount and types of resources at the right time to perform required tasks. MSS had a serious problem to meet its own promise to deliver a set of artefacts by the end of July with only 4 staff (Section 4.4.1). The available resource level proved to be inadequate. It was only through a "re-planning" exercise did the project team realise the scale of the problem. A new supplier had to be brought in as a matter of emergency. The resource level requested in the new plan (see Table 4.3) was considered to be excessive (SLC project manager suspected it to be excessive to the tune of 20% at the time – private conversation). Even with such seemingly generous resource availability, and assisted with significant scope reduction, the eventual delay of Release 1 delivery was significant.

Long-term resource planning might be difficult due to unforeseeable events. How about the accuracy of short-term planning? From November 2000 to February 2001, the project team was focusing on short-term objectives. But the record of assessing what had to be done and when the objectives could be achieved was not an impressive one (Section 4.4). The project could not predict beyond a week or two as for what could be achieved. There could be many potential explanations behind repeated failures to meet delivery targets. Among other things, the client's pressure to have a working system delivered before Christmas could have jeopardised the project team's sense of judgement before Christmas. More likely, however, is that software development is full of uncertainty that defies careful planning and estimating.

4.7.7 Risk Management

Risk management was much emphasised in the project. A risk management process was in place (SLC, 38). Risks were logged and assessed with pre-emptive actions documented. But implementing the intended actions was often a challenge. For example, there was a risk registered (R005)[13] as early as 08/06/2000 regarding "timely and cost effective delivery" from third party suppliers (SLC, 1, 14). The proposed action was to have "[c]lear supplier responsibilities identified and enshrined in contract" and "Supplier Management responsibilities clearly identified" (SLC, 14). Though efforts were made toward that direction, the risk manifested itself as a serious issue by the end of August 2000 regarding MSS' deliverables. It was not clear how the risk management could have made any difference. The problem seemed to be that when the risk had manifested itself, there were few options available due to the constraints of time and cost. To resolve the issue of MSS' inability to deliver, SLC had to bring another supplier in a hurry (Section 4.3.4) at a considerable extra expense borne by the client.

---

[13] R005 was the number assigned to the risk in the risk register. The risk number was sequential for the convenience of referencing. The same applies to the other risk numbers mentioned below.

4.7.8 Project Success or Failure?

The WebShop Project was not breaking any new technological ground. By the Year 2000, there were many sophisticated retail websites (such as www.amazon.com) available on the Internet. Yet the project was full of challenges. As commented in the beginning of this Section, it rightly belongs to the "challenged" category since it was neither an outright success, nor a cancelled project before Release 1.

The WebShop Project serves to illustrate many problems that ISD projects face. The definitions of activities and processes never stood still. There were serious uncertainties with the development activities. The problem the client faced was that there had seemed no alternative to delaying the project in the first place and accepting a product that was unsatisfactory in the end.

As a postscript, it should be noted that, within four months of the website going live on the Internet, UKOne decided to sell off its Retail operation. The retail website was withdrawn from the Internet before the retail operation's ownership was changed. The hardware from the project was recycled for other uses and the application software was scrapped apart from one webpage retained as a link to some other services provided by UKOne.

4.8 The WebShop Project Chronology

22	March	2000:	UKOne Board confirmed its e-commerce vision.
20	April	2000:	WebShop Project Phase 0 was initiated.
April-May		2000:	Workshops conducted to gather business requirements.
12	May	2000:	Attempt to finalise commercial agreement with MSS.
01	June	2000:	NQ made a presentation.
08	June	2000:	NQ issued a proposal.

09 June 2000: NQ had been selected as the design partner.

23 June 2000: Contract with NQ signed.

26 June 2000: Technology decision made in favour of Microsoft and thus MSS.

27 June 2000: NQ not comfortable with the new Microsoft Commerce Server 2000.

28 June 2000: MSS volunteered to develop main components within four weeks. MSS' tasks expanded and NQ's tasks shrunk.

04 July 2000: MSS' estimated effort was 273.5 days.

14 July 2000: Responsible areas between MSS and NQ agreed and documented.

27 July 2000: SLC presentation to SubA not accepting a fixed price as requested.

01 August 2000: Demonstration of Retail and Customer Services prototype.

02 August 2000: MSS indicated that insufficient information for development.

04 August 2000: MSS requested a project audit by an external project manager.

11 August 2000: MSS internal review indicated a minimum 306 days effort for development alone.

14 August 2000: MSS initiated project audit feedback recommending the preparation of a full feature-set.

18 August 2000: Feature set matrix indicating 505 days development effort.

21 August 2000: Feature set matrix reviewed. A possible scenario with scope reduction required 320 days development.

22 August 2000: MSS was unwilling to commit to development for less than 430 days

23 August 2000: Client review recommended scope reduction to 311 days according to the feature set matrix. SLC considered additional third party developers.

24 August 2000: In preparing a draft project plan, MSS identified additional tasks. Total estimate went up to 528 days.

25 August 2000: MSS produced revised plan showing 506 days.

25 August 2000: SLC reviewed other Microsoft partners for extra development

capacity.

End of August 2000: IFG was brought into the project to provide extra developers.

05 September 2000: MSS delivered a draft architecture document.

18 September 2000: Project deliverables split into Release 1 and Release 2.

30 October 2000: Microsoft Commerce Server 2000 not released as planned.

13 November 2000: Commerce Server 2000 released to manufacturing.

13 December 2000: Release 1 launched internally.

09 January 2001: Prepared to launch Release 1 to Internet within two weeks.

26 February 2001: Release 1 launched to the Internet.

13 March 2001: The project was terminated. Release 2 was to be developed as a separate project.

4.9 Internal References

Reference#	Reference Source	Comments
IFG, 1	Hours - first chargeable period (to 14sep00).msg (content dated 29/11/2001)	Report from IFG to SLC Regarding Hours Worked
IFG, 2	Hours Charged 11th to 17th November.rtf (content dated 21/11/2001)	Report from IFG to SLC Regarding Hours Worked
MSS, 1	Roadmap.doc (file dated 6/7/2000)	MSS' Plan of Actions
MSS, 2	status report 240800.doc (content dated 24/8/2000)	MSS Report on WebShop Project Status
MSS, 3	Architecture Integration.doc (content dated 1/9/2000)	MSS' Draft Architecture Design
MSS, 4	Consolidated Report 20-Nov-2000.doc (content dated 20/11/2000)	MSS Weekly Project Report
MSS, 5	findings3.doc (content dated 11/08/2000)	MSS Review of WebShop Project
MSS, 6	DS feature set 2_ 2208 #s.xls (content dated 22/08/2000)	WebShop Featured List
MSS, 7	Consildated Report 06-Nov-2000.doc (content dated 06/11/2000)	MSS Weekly Project Report
MSS, 8	Consildated Report 5-10-00.doc (content dated 05/10/2000)	MSS Weekly Project Report
NQ, 1	Nqcv1.ppt (file dated 01/06/2000)	NQ Sales Presentation
NQ, 2	NQ propv1.doc (file dated 08/06/2000)	NQ Sales Proposal

Reference#	Reference Source	Comments
NQ, 3	Working~Process~d1.doc (file dated 24/7/2000)	Working process agreement between NQ and MSS
NQ, 4	NQ Final Contract.doc (file dated 16/06/2000)	Proposed Contract Between NQ and SLC
SLC, 1	SLC_pqp_v1.2 working.doc (file dated 5/10/2000)	SLC Project Quality Plan (version1.2)
SLC, 2	e-bus weekly report 000602.doc (content dated 2/6/2000)	Weekly Project Report to Project Sponsor
SLC, 3	weekly0906sl.doc (content dated 9/6/2000)	Weekly Project Report to Project Sponsor
SLC, 4	weekly1606bg.doc (content dated 15/6/2000)	Weekly Project Report to Project Sponsor
SLC, 5	weekly2306bg.doc (content dated 23/6/2000)	Weekly Project Report to Project Sponsor
SLC, 6	technology decision.doc (file dated 23/6/2000)	Pros and Cons of Technology Platforms
SLC, 7	EBusquidIPFv3.doc (file dated 15/6/2000)	Initial Project Form Agreed between UKOne and SLC
SLC, 8	Proj_Presentation_130301.ppt (content dated 13/3/2001)	Weekly Project Presentation to the Project Steering Group
SLC, 9	KeyMilestones3006.ppt (file dated 30/06/2000)	Outline Project Milestones
SLC, 10	KeyMilestones0607.ppt (file dated 06/07/2000)	Outline Project Milestones
SLC, 11	Proj_Presentation_1809.ppt (content dated 18/9/2000)	Weekly Project Presentation to the Project Steering Group

Reference#	Reference Source	Comments
SLC, 12	weekly2306sl.doc (content dated 23/6/2000)	Weekly Project Report to Project Sponsor
SLC, 13	weekly0707sl.doc (content dated 07/07/2000)	Weekly Project Report to Project Sponsor
SLC, 14	riskreg(2).doc (content dated 3/11/2000)	Project Risk Register
SLC, 15	Projcosts- workingv40.xls (file dated 5/6/2000)	Project Cost Spreadsheet
SLC, 16	Projcosts- workingv60.xls (file dated 31/7/2000)	Project Cost Spreadsheet
SLC, 17	weekly0109sl.doc (content dated 1/9/2000)	Weekly Project Report to Project Sponsor
SLC, 18	Proj_Presentation_0210.ppt (content dated 02/10/2000)	Weekly Project Presentation to the Project Steering Group
SLC, 19	Milestones 1809.ppt (file dated 15/9/2000)	Outline Project Milestones
SLC, 20	Proj_Presentation_0711.ppt (content dated 7/11/2000)	Weekly Project Presentation to the Project Steering Group
SLC, 21	Proj_Presentation_090101.ppt (content dated 09/01/2001)	Weekly Project Presentation to the Project Steering Group
SLC, 22	Proj_Presentation_230101.ppt (content dated 23/01/2001)	Weekly Project Presentation to the Project Steering Group
SLC, 23	Proj_Presentation_300101.ppt (content dated 30/01/2001)	Weekly Project Presentation to the Project Steering Group
SLC, 24	SLC_pqp_v1.0.doc (file dated 30/6/2000)	SLC Project Quality Plan (version1.0)

Reference#	Reference Source	Comments
SLC, 25	feature set 2_ 210800 (97).xls (content dated 21/8/2000)	Project Scope Feature List
SLC, 26	Proj_Presentation_0509.ppt (content dated 5/9/2000)	Weekly Project Presentation to the Project Steering Group
SLC, 27	Proj_Presentation_1010.ppt (content dated 10/10/2000)	Weekly Project Presentation to the Project Steering Group
SLC, 28	Proj_Presentation_1710.ppt (content dated 17/10/2000)	Weekly Project Presentation to the Project Steering Group
SLC, 29	JBcosts audit trail.xls (file dated 17/07/2000)	Project Cost Spreadsheet
SLC, 30	varicont008.doc (file dated 17/10/2000)	Project Scope Variation
SLC, 31	Proj_Presentation_1912.ppt (content dated 19/12/2000)	Weekly Project Presentation to the Project Steering Group
SLC, 32	Proj_Presentation_2410.ppt (content dated 24/10/2000)	Weekly Project Presentation to the Project Steering Group
SLC, 33	testing_strategy1.1.doc (content dated 16/8/2000)	Project Testing Strategy
SLC, 34	Proj_Presentation_3110.ppt (content dated 31/10/2000)	Weekly Project Presentation to the Project Steering Group
SLC, 35	Proj_Presentation_2411.ppt (content dated 24/11/2000)	Weekly Project Presentation to the Project Steering Group
SLC, 36	weekly1205.doc (content dated 12/05/2000)	Weekly Project Report to Project Sponsor
SLC, 37	WebShop Project Technical Report v0.2.doc (file dated 31/07/2000)	SLC Management Review of WebShop Project

Reference#	Reference Source	Comments
SLC, 38	risk_process.doc (file dated 26/06/2000)	Project Risk Management Process
SLC, 39	NQ_SLC_Bus_review_d9_jb.xls (file dated 18/08/2000)	WebShop Front-end Review Record
SLC, 40	weekly2209bg.doc (content dated 22/09/2000)	Weekly Project Report to Project Sponsor
SLC, 41	weekly2010bg.doc (content dated 20/10/2000)	Weekly Project Report to Project Sponsor
SLC, 42	variation status.doc (file dated 08/01/2001)	Summary of Project Variations
SLC, 43	Quality Criteria agenda briefing docs.doc (file dated 06/10/2000)	Quality Criteria Workshop Agenda and Brief
SLC, 44	Draft_Quality_criteria.doc (file dated 20/10/2000)	Draft Quality Criteria
SLC, 45	Draft_Quality_criteriav2.doc (file dated 16/02/2001)	Draft Quality Criteria
SLC, 46	www CSF.doc (file dated 23/02/2001)	Project Critical Success Factors
SLC, 47	weekly0408sl.doc (file dated 04/08/2000)	SLC Weekly Report
SLC, 48	weekly1108bg.txt (file dated 14/08/2000)	SLC Weekly Report
SLC, 49	weekly1808bg.doc (file dated 18/08/2000)	SLC Weekly Report
SLC, 50	Proj_Presentation_260201.ppt (content dated 26/02/2001)	Weekly Project Presentation to the Project Steering Group

Reference#	Reference Source	Comments
SubA, 1	Capture_Spec_v6.doc (content dated 27/06/2000)	Online Customer Capture Specification (version 6.0)
SubA, 2	Energy Servicing V6.0.doc (file dated 30/06/2000)	Energy Servicing Specification (version 6.0)
SubA, 3	MS v_6.doc (file dated 04/08/2000)	Marketing Functional Specification (version 6.0)
UKOne, 1	UKOne Annual Report & Accounts/Form 20-F 2000/01	UKOne Annual Report for the Year 2000/2001[14]

[14] Annual reports are published documents but to keep the client's identity confidential in line with the research methodology advocated in Chapter 2, the corporation's name is disguised here and the document is treated as an internal one.

Chapter 5: Case 2 – The Gas Meter Reading Management System

(March 2001 - October 2001)

5.1 Background

Historically, the gas market was a monopoly run by BritishGas (Beesley et al., 1994; O'Neill, 1996). Even though it was privatised in 1986, BritishGas continued being a monopoly player in the gas transportation and gas supply market (MMC, 1993). Since 1996, competition has been gradually introduced into the residential gas supply market (Giulietti and Price, 2000). By 1998, the gas supply market was "fully liberalised", which meant that all gas suppliers can sell to customers anywhere in the UK (Ofgem, 1998).

Transco, as part of BritishGas, used to act as the meter reading agency (MRA) for gas suppliers as well as managing the gas meters as part of the gas transportation network. Companies entering into the gas supply market theoretically had a choice between using Transco or a licensed company for obtaining meter readings. In reality, however, competition in this part of the market was not happening. Most companies without any readily available alternative simply chose to "do nothing" and have kept Transco as their MRA (Ofgem, 1998). Even those companies with alternative means of obtaining meter readings took only limited steps to "unbundle" the meter reading business from Transco due to other considerations like system constraints and geographic coverage. For example, SubB, the subsidiary of UKOne responsible for its gas operations in the open gas market, unbundled less than 50% of meter reading business from Transco by early 2001 even though it operated its own meter reading services. Over 50% of SubB's meter reading business remained with Transco.

Indeed, Transco was in a privileged position to act as an MRA for gas suppliers. Transco performed the upper stream function as a Public Gas Transporter (PGT) while owning and

maintaining gas meters. There was a single integrated software system (known as Sites and Meters) in Transco handling information regarding the meters and meter readings. This set-up gave Transco a commercial advantage no other competitor could have and was deemed to be "discriminatory" (*Ofgem*, 1998).

As a result, in October 1998 Ofgem published its first proposal on "Securing effective competition in the gas metering and meter reading services" (Ofgem, 1998). It aimed to bring competition into the Non Daily Meter (NDM) part of the meter reading market, which was most labour intensive, and thus costly. After various consultation stages, Ofgem published its final proposal in May 2000 (Ofgem, 2000c) setting April 2001 as the deadline for:

> "Implementation of stand-alone NDM reading IT system, and removal of Transco's meter reading business' access to sites and meters database..." (p. 60)

This came to be known in the gas industry as MRA/PGT Separation Project, referred to as MPS Project throughout this case study. The "stand-alone NDM reading IT system" that Transco was to acquire and implement was called Proteus, which implemented a whole set of new exchange file formats. Gas suppliers were required to comply with the changes or lose their operating licences. The task of managing this regulatory requirement for UKOne was undertaken by its subsidiary SubB that was responsible for gas operations.

5.2 SubB's Requirements

SubB had acquired a series of systems in support of its gas operations. MPS Project had an impact on many of them. Figure 5.1 outlines the existing systems and how they were related. All systems apart from "Transco PGT/MRA" were owned and operated by SubB. The core ones were (all system names have been slightly altered from the original ones for anonymity):

- SPMS: Supplier Point Management System
- UMRS: Unbundled Meter Reading System
- GDCS: Gas Data Collection System

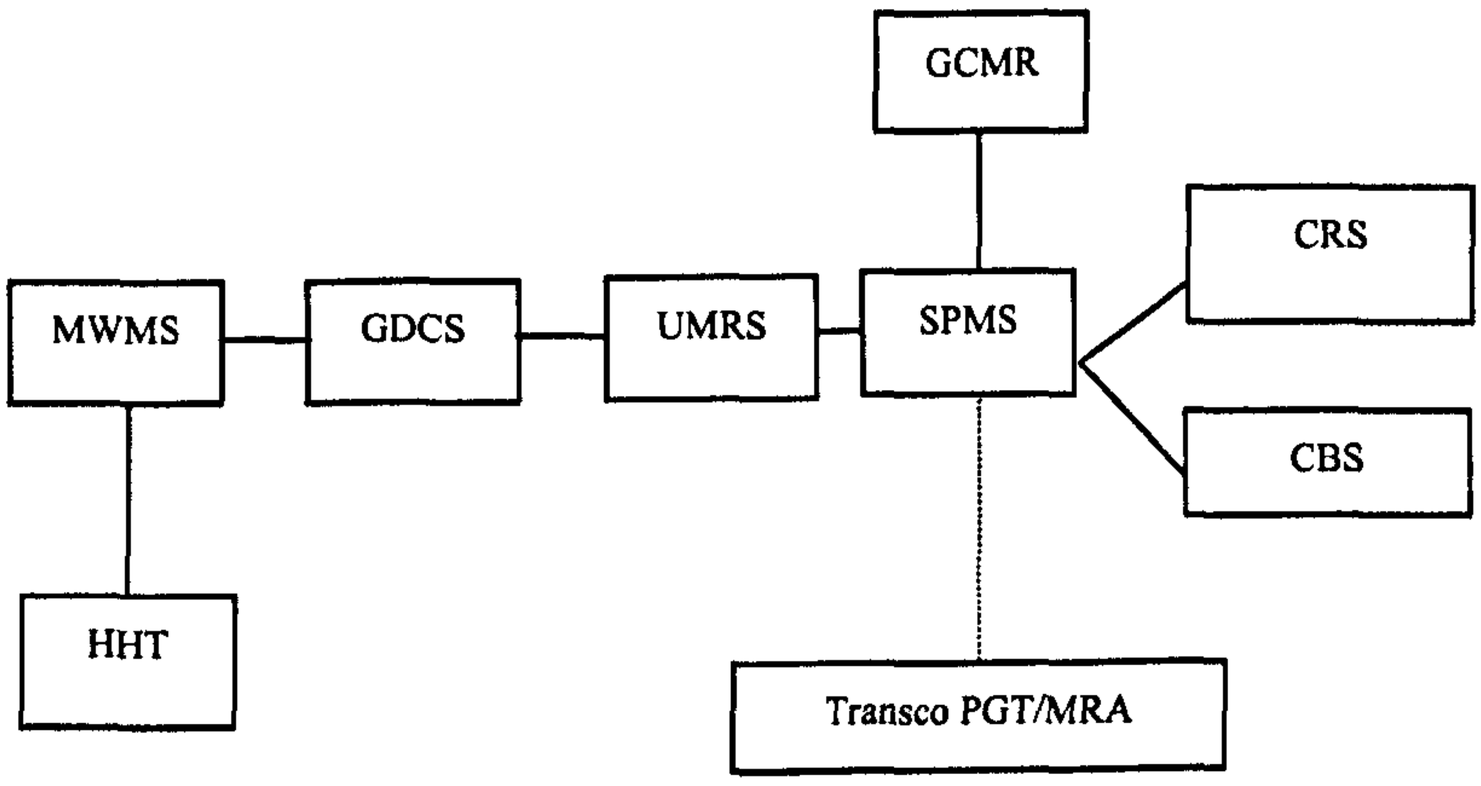


Figure 5.1: Systems related to gas operations in SubB before the MPS Project

Other systems were peripheral and related to the above systems as shown in Figure 5.1:

- GCMR: Gas Customer Meter Reading
- MWMS: Meter Work Management System
- HHT: Hand-Held Terminal for collecting meter readings in the field
- CRS: Customer Registration Systems
- CBS: Customer Billing Systems

SPMS was at the heart of SubB's gas operations. It maintained SubB' gas customers portfolio, all records of supply points and meter points plus all historical meter reading records. It was the gateway to communicate with Transco PGT for gains and losses of gas customers. It was also the gateway through which Transco MRA had been supplying the "bundled" meter readings to SubB. Furthermore, it was used for SubB to communicate their unbundled meter readings to Transco PGT for consumption reconciliation. All these would have to be changed since SPMS was not able to work with the new Proteus file formats (SLC, 2, 5).

As mentioned in Section 5.1, SubB carried out part of the gas meter reading itself. The process as it was can be briefly described as follows. The readings were first collected in the field by HHT. MWMS retrieved the readings from HHT and passes them onto UMRS for

processing. The readings were then recorded by SPMS for further communication to Transco PGT for authorisation. After authorisation was obtained, SPMS passed them onto customer billing systems (CBS). Customer registration system (CRS) would also be affected if a new system was to be implemented to accept new customer records. The key SubB's requirement was that the gas systems were either modified or replaced so that SubB could work with Transco's Proteus file formats for meter readings exchanges, validation, confirmation and error handling etc. Depending on changes related to the core systems, the peripheral systems might had to be adapted accordingly.

5.3 Options Available and Supplier Selection

Although as early as May 2000, it was known in the industry that the deadline for MPS Project was to be April 2001, it was not until early January 2001 when SubB's management commissioned SLC to carry out a feasibility study (SLC, 1). SLC duly completed the feasibility study (SLC, 1), which found that:

- The timescale was desperately short. However, the consolation was that, according to industry sources and the IT supplier community, no one was ready for the April 2001 deadline. The new implementation date was likely to be July 2001;
- SLC would not be prepared to undertake the whole project alone in such a short timescale. The risk was considered too high.

The recommendation was to acquire an application software package. There were two potential suppliers considered. One was a supplier called AGL. AGL had started developing a software product to cater for MRA/PGT Separation with nine gas suppliers already. Although AGL had a good reputation in the IT industry, and also had developed a good working relationship with both SubB and SLC, AGL's solution required the replacement of both SPMS and UMRS. Replacing SPMS would have meant "significant additional systems analysis and systems integration tasks" (SLC, 1), thus more expensive and more risky.

The other potential supplier was AES. Like AGL, AES claimed that they had started developing systems for MRA/PGT Separation, although with only two gas suppliers already signed up at the time. AES's solution was for SubB to keep SPMS and only replace UMRS with a new "module" called "Gas Meter Reading Management" (GMRM) to manage Proteus file formats. Since AES had supplied SPMS in the first place, it was considered to be of lower risk in integrating GMRM with SPMS. However, AES' track record of delivery, quality and services had been known to be poor (SLC, 1). The feasibility study concluded that all options carried high risks. The cost implication of the two options is summarised in Table 5.1 (based on SLC, 1, 19), which shows a small advantage in favour of AES. The feasibility study recommended AES "WITH SEVERE RESERVATIONS" (SLC, 1).

Table 5.1: Indicative cost summary of the two supplier options

	AGL	AES
Software Application License Cost	£125K	£250K
Supplier Consultancy for System Integration & Implementation	£100K	£100K
Annual Operational and Support Cost Charged by Supplier	£175K	£50K
Other Items (including acquiring other system elements, system integration and implementation etc)	£1000K	£841K
Total Indicative Costs	£1,400K	£1,241K

By March 2001, the project was under tremendous pressure to get started. SubB finally decided to commission SLC as the prime contractor and opted for AES as the software supplier. One of the key factors that persuaded SubB to choose AES was the fact that there would be "minimal" changes to SPMS (SLC, 1). The proposed solution was to (SLC, 2):

- Keep SPMS as the key link to Transco PGT, Customer Registrations and Customer Billing Systems;
- Replace UMRS with GMRM to manage meter readings from Transco MRS and SubB's own systems;
- Re-write GDCS to be GDCS2 to be compatible with Proteus file formats.

There were other associated requirements to be met, for example (SLC, 2):

- Data extraction from SPMS/UMRS and data population into GMRM;
- Data migration from GDCS to GDCS2;
- Management information reports from GMRM;
- Development of a web based customer reads interface;
- Modifying GCMR so that the customer reads were fed into GMRM instead of SPMS.

The targeted architecture is shown in Figure 5.2.

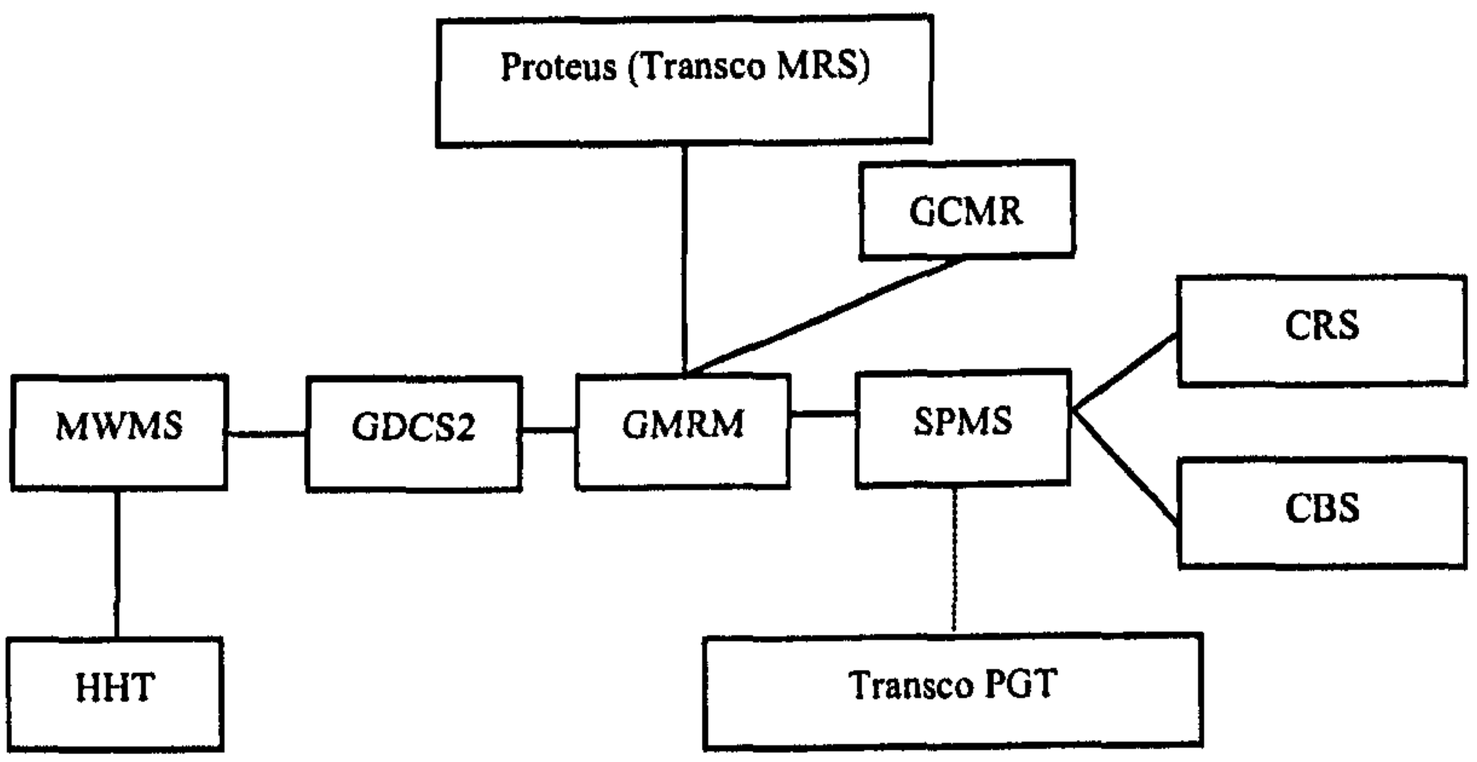


Figure 5.2: Simplified architecture for the proposed solution

The whole project was complex with a number of workstreams involving both external suppliers and internal developers in SLC. The focus of this case study is on the part of the project that was to acquire GMRM from the external supplier AES.

5.4 Project Organisation and Resources

SLC took on the role as the prime contractor assigning one of its senior managers to be the MPS Project Manager. Most of the other project team members were also from SLC. A senior manager within SubB acted as the Project Sponsor. SubB also allocated internal resources with business domain knowledge working with SLC project team. The project organisation chart is shown in Figure 5.3 (based on SLC, 2). The Project Manager had responsibility for all aspects of the project. Apart from reporting into the Project Sponsor, the Project Manager reported regularly to SLC management team. The other project team members, though they

had their own respective line managers, were managed directly and solely by the Project Manager for the duration of the project.

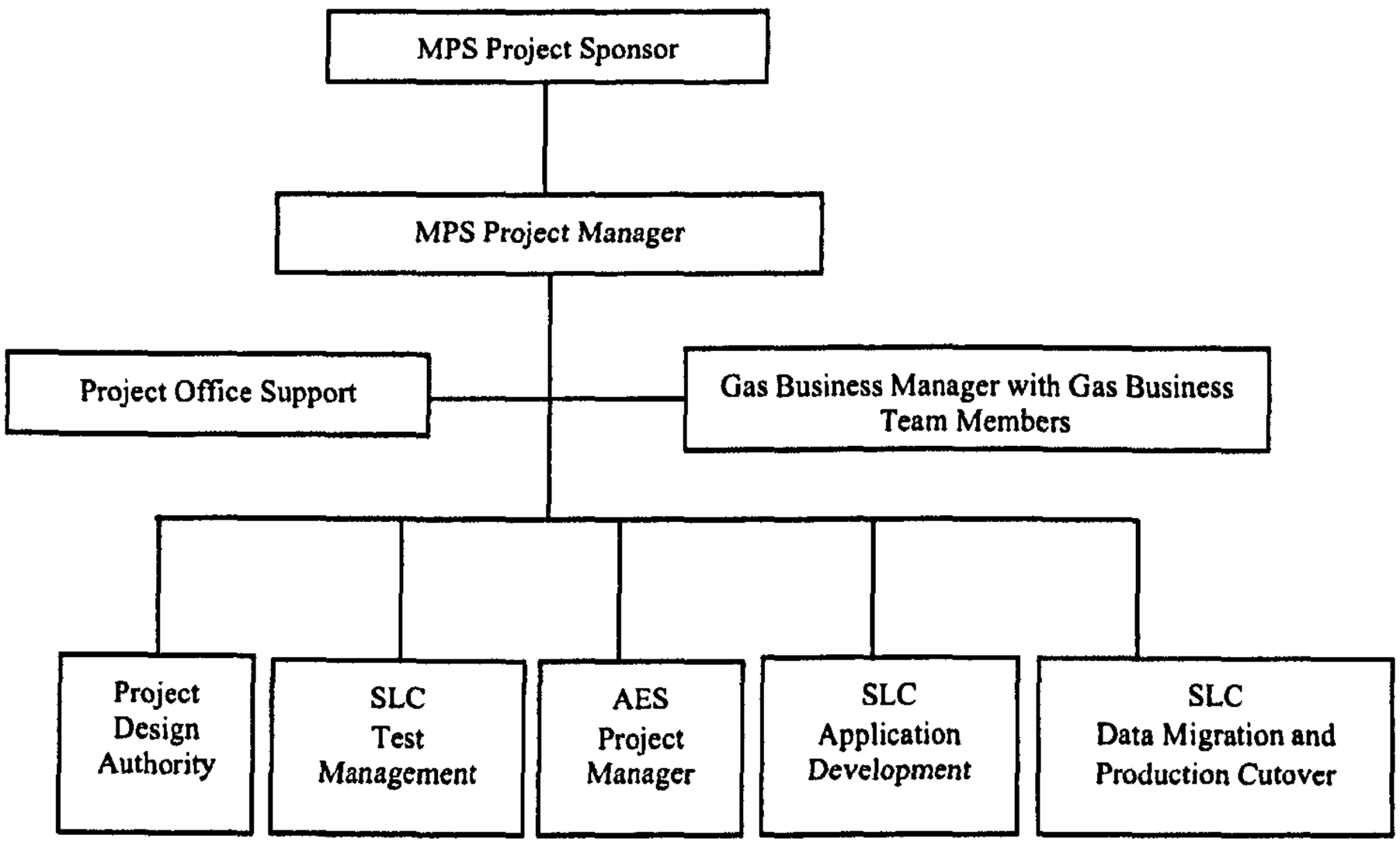


Figure 5.3: The MPS project organisation

The appointed Project Manager had over at least 20 years of IT project-related experience with a track record in managing complex and high profile projects. He was politically astute and good at communication with the client, the project team and the suppliers. The project manager also prided himself for the fact that the client entrusted him to be *the* project manager, i.e. there was to be no project manager from the client to oversee the project progress on a day-to-day basis.

5.5 Contracting with AES

The contract for GMRM was between UKOne (rather than SubB or SLC) and AES (AES, 4). The contract was for AES to supply GMRM for a fixed fee. Observations are made here regarding the contract, including deliverables and the delivery schedule, price and contract administration.

5.5.1 Deliverables and Delivery Time

The deliverables and milestones were found in AES' contract proposals. AES made the initial commitments in the proposal dated 28/03/2001 as shown in Table 5.2 (based on AES, 2).

Table 5.2: AES' deliverables schedule dated 28/03/2001[15]

Deliverables	Date
Order to be placed and accepted by AES	02/04/2001
GMRM Release 1 Development Complete	30/04/2001
GMRM Release 1 Delivery to SLC with Installation and User Guide	30/04/2001
GMRM Release 1 On-site Acceptance Test Complete	04/05/2001
GMRM Release 2 Development Complete	02/07/2001
GMRM Release 2 Delivery to SLC with Installation and User Guide	02/07/2001
GMRM Release 2 On-site Acceptance Test Complete	04/07/2001
Client Raise priority 1 and 2 PIRs or sign acceptance certificate	04/07/2001
AES Deliver the fixes for priority 1 and 2 PIRs	24/07/2001
Client re-test priority 1 and 2 PIRs fixes complete	31/07/2001
Client sign acceptance certificate	31/07/2001
Start of Warranty Period	31/07/2001

The exact features of Release 1 and Release 2 were not detailed in the proposal. It was understood, however, that Release 1 was to provide the core functionality that would allow SubB to participate in an industry testing in June. Release 2 was expected to have extra functionality built in to manage multiple MRAs and to manage ad-hoc reads, which were required to meet SubB's operational needs at the time of implementation in July. The project team considered Release 2 delivery on 02/07/2001 being too late if the release was to be properly tested and to meet the industry deadline of 23/07/2001. This concern was taken up by AES and Release 2 delivery date was moved to 22/06/2001 in the revised proposal (AES, 3,4).

5.5.2 Contract Price and Payment Schedule

At the time of the feasibility study, AES' proposed license fee was £250K (SLC, 1). At the time of project initiation at the end of March, the license fee was requested to be £350K

[15] The term "Phase" was used instead of "Release"" in the original document (AES, 2). "Release" was used in later documents (AES, 3, 4). To be consistent, the term "Release" is adopted throughout the case study.

(AES, 2). The increase was apparently due to a mistaken calculation. It was eventually negotiated down to £300K (AES, 3). The payment schedule was requested by AES (AES, 3) as shown in Table 5.3 (based on AES, 3).

Table 5.3: AES' initial payment schedule

Project Milestone	Price (ex VAT)	Indicative Dates per Proposed Project Plan
Order Acceptance	50% (£150K)	2/4/2001
Release 1 Delivery	15% (£45K)	30/4/2001
Release 2 Delivery	15% (£45K)	22/6/2001
Final Acceptance of Release 2 delivery	20% (£60K)	23/7/2001

5.5.3 Acceptance Criteria

There had been no product acceptance criteria set by the client before the project started. AES suggested in its project proposal (AES, 3, p. 20) that the software be deemed as accepted should any of the following occur:

- SubB or SLC sign an Acceptance Certificate following formal Acceptance Testing;
- SubB use the Software in live operation;
- 30 days have elapsed from completion of the Acceptance Testing.

An email from the client's contract manager dated 25/07/2001 (SubB, 1) made some further conditions regarding software acceptance in relation to a final payment of £25K that was held back. The items related to GMRM product are listed below:

- Delivery of AES' Site Acceptance Test Report and an action plan for the resolution of reported GMRM defects;
- Delivery of an agreed list of GMRM defects at the conclusion of the UKOne/SLC System Acceptance Test and an action plan for their resolution (This test was to follow AES' Site Acceptance Test within three weeks);
- Delivery of all agreed GMRM documentation as outlined in the proposal with a plan for any updated versions to incorporate any corrections/omissions identified by UKOne/SLC raised prior to the end of warranty.

No provision seemed to have been made in any of available contractual documents regarding

the possibility of the software product not being accepted or not operating to the client's satisfaction.

5.5.4 Contract Administration

There are two observations regarding the contract administration. The first is related to the contractual commitment between the client and the supplier. The contract was effective from 03/04/2001 but was not signed until after 13/06/2001 (AES, 5). The fact that AES started the delivery effort before the contract was actually signed suggests that UKOne must have made a contractual commitment to AES in some way, formal or informal. Exactly how that commitment was made is not clear. The rush to making a contractual commitment without having a contract signed was partly due to the fixed deadline being imposed externally, partly due to the time required to agree the details of the contract and associated proposals. Such an action to commit resources without a signed contract could also be interpreted as a sign of trust and partnership. Certainly AES did commit to work as "partners" with UKOne in its proposals (AES, 2, 3). Though this is difficult to interpret since the term "partner" was perhaps used loosely without any legally defined meaning. The actual lack of trust was made quite clear later in a threat issued by AES for holding back Release 2 should there be further delays to signing the contract (AES, 5).

The second observation is that the quality control and product acceptance procedure was not specified in any detail. Under the heading of "Details of, Procedure for and Commencement Date of Acceptance Tests", all that was stated is:

> "25th June 2001 Acceptance testing commences, 23rd July 2001 Acceptance Certificate signed" (AES, 4)

This is hardly detailed! No provision was made regarding the possible outcome of the software product NOT being accepted.

5.6 Project Budget and Cost Control

While the license cost of GMRM from AES was "fixed" from the early stage of the project, other project cost items were changeable depending on the project progress. The main changeable items were costs for project consultancy provided by AES and system testing, integration and implementation activities carried out by SLC. Table 5.4 gives a breakdown of the MPS Project total cost as seen from mid July to mid September 2001.

Table 5.4: MPS Project cost estimation and monitoring

	Initial Budget (SLC, 2)	MPS Project Cost Estimation and Monitoring					
		18/07/2001 (SLC, 21)	07/08/2001 (SLC, 22)	16/08/2001 (SLC, 23)	22/08/2001 (SLC, 24)	27/08/2001 (SLC, 25)	13/09/2001 (SLC, 26)
SLC Resource Total	549,500[16]	603,575	608,051	593,581	615,348	710,440	765,177
AES Resource Total	142,000	115,000	90,542	90,542	90,542	98,887	103,887
AES-GMRM Licence	300,000	300,000	300,000	300,000	300,000	300,000	300,000
AES-GMRM Support	40,000	40,000	40,000	40,000	40,000	40,000	40,000
Other Costs	269,000	203,640	220,078	220,078	220,078	220,078	220,078
Project Total	1,300,500	1,262,215	1,258,670	1,244,201	1,265,967	1,369,405	1,429,142

Interestingly, AES consultancy expenditure came under budget. According to a private conversation with the project manager, this was because the project team and SubB management demanded free resources from AES to resolve quality problems in the process of testing and implementing GMRM. The biggest overspend was on the SLC resources. The original budget for SLC resources was £549,500, which was later increased, partly due to some cost variations and partly due to applying 6.3% unit (man-day) price increase by SLC from April 2001. The target budget was thus around £614K for SLC (SLC, 20). By mid August 2001, projected SLC resource cost was within the budget level. The cost estimate went just over the budget one week later and then went up quickly by over £100K less than two week later, an increase of nearly 20%. Two weeks later, the cost estimate went up further by over £50K or an additional 8%. These increases alone were responsible for the overall

[16] This figure was later increased, partly due to some cost variations and partly due to applying a 6.3% man-day price increase. The target budget for this item in July 2001 was thus around £614K (SLC, 20).

project cost overrun of 10%, even though other cost items had come on or under budget.

The project was initially planned to be completed by 17/08/2001 (SLC, 8). That was not achieved because GMRM did not function as required. AES had to continue the software development and SLC had to retain a number of staff for further testing and system integration. The project could not be closed down since GMRM was not stable in the production environment, requiring close manual intervention and alternative arrangements (data converters) to deal with daily data processing. The project had to be extended for more than two months, officially closed on 26/10/2001 (see Section 5.8), causing a considerable amount of cost overrun. The cost was passed onto SubB since the overrun was caused by AES. SLC was there to help recover a disastrous situation and thus expected to be paid in full.

5.7 Product Quality Control and Acceptance Testing

AES provided a set of acceptance testing scenario and scripts (AES, 7) with the first GMRM release. There were 28 test scenarios that the core GMRM product should be able to perform. AES advised that the first release could only perform 10 of these scenarios (SLC, 18). Even with the 10 scenarios, and using artificially constructed test data with nine Metering Point Administration Numbers (MPANs)[17] (AES, 13) supplied by AES, there were seven defects raised with issues ranging from test scripts errors to failures of data import or export (SLC, 18). The client representatives provided nine comments on the graphical user interface design with the awareness that the development had not been completed at that stage (SLC, 18). Unfortunately, the pattern of software release, acceptance testing and defect communications was repeated many times toward the system implementation date of 23/07/2001 and thereafter (SLC, 13, 14, 15, 16). As a result, no acceptance certificate could be signed either by SubB or SLC, as initially envisaged. As of 29/07/2001, SubB held back £25K payment, the release of

[17] Each MPAN is a point where meter reading is taken. A single customer may have one or more MPANs associated.

which was subject to a number of conditions including a further round of acceptance testing (SubB, 1; Section 5.5.3).

5.8 Actual Project Progresses

The actual software releases are shown in Table 5.5.

Table 5.5: GMRM delivery progress as compiled on 26/10/2001

Release#	Received Date	Notes
GMRM1.00.01	30/04/2001	
GMRM1.01.00	23/05/2001	This was to be delivered on 21/05/2001
GMRM2.00.00	27/06/2001	The release only contains core functionality as defined for Release 1.
GMRM2.01.00	13/07/2001	This release had to be implemented on 23/07/2001 with limited functionality.
GMRM2.01.01	25/07/2001	This and following patches (small releases) were for fixing various defects in GMRM2.01.00.
GMRM2.01.03	27/07/2001	
GMRM2.01.04	02/08/2001	
GMRM2.01.05	07/08/2001	
GMRM2.01.06	09/08/2001	
GMRM2.01.07	13/08/2001	
GMRM2.01.08	03/09/2001	
GMRM2.01.09	04/09/2001	
GMRM2.02.00	14/09/2001	Release 2 as originally defined. The Release included extra functions which had to go extensive testing[18]

On 26/10/2001, SLC closed its project officially. The product was handed over to production support with the project team re-assigned to support activities. At that time, the production system had Release GMRM2.01.08 that performed two of six major business processes:

- Gain (live but producing exceptions which must be manually resolved);
- Loss (live but producing exceptions which must be manually resolved);
- Reads processing (not working);
- Meter Exchange details (not working);
- Multiple MRA function (not available);
- Ad-hoc Read (not available).

No official product acceptance certificate was ever signed. However, according to one of the

[18] Based on conversations with colleagues continued to be associated with GMRM, SubB decided to implement a new application from a different supplier toward the end of 2002.

AES' initial acceptance criteria, the product would have deemed to be accepted if the client used it for "live production" (Section 5.5.3).

5.9 Project Issues and Within Case Analysis

The following issues are discussed in this section: supplier selection, system architecture, data integration, documentation, risk management, and post-implementation review. A brief comment is made regarding the success and failure of the GMRM product and the MPS Project as a whole.

5.9.1 Supplier Selection

The selection of AES was a considered decision as a result of SLC's feasibility study, though with "WITH SEVERE RESERVATIONS" (Section 5.3). In other words, SubB knew it was taking high risks by choosing AES. Yet comparing with other options, the risks associated with AES were considered to be more manageable. There is no evidence from the project to indicate that the initial decision was necessarily wrong, even though the software product from AES was far from adequate. There was simply no better options.

5.9.2 System Architecture

In AES' initial proposal, there was a promise of close integration of GMRM and the existing system SPMS. In assessing various options, this tight integration helped the supplier selection decision to favour AES significantly. Yet GMRM as delivered to SubB stood on its own in managing the customer portfolio. In other words, GMRM was not "closely" integrated with SPMS. The SLC project team raised a question to AES Programme Manager about the architectural change compared with the original understanding. The question was why GMRM had been designed to be separate from SPMS in terms of portfolio management instead of being integrated with SPMS as initially understood to be. After all, SPMS already had a well-established customer portfolio and a tried-and-tested process to maintain it. The

fact GMRM was stand-alone meant that SubB had introduced another system that had a duplicate portfolio to be maintained and the new portfolio must be synchronised with the one in SPMS. A few weeks after GMRM was implemented, there were problems in achieving such a synchronisation.

AES' answer to the above question was that GMRM had been designed as a stand-alone system so that it could be sold to a new client without SPMS. Otherwise, GMRM could only be sold to those already with SPMS installed or sold to new clients with SPMS bundled together. Neither option was going to be attractive to AES since the existing SPMS client base was small and SPMS had become too dated to be sold to new clients. AES' decision seemed perfectly rational to AES, but it created a heavy burden for SubB. Supporting two systems was more costly. What was worse was the potential problem of data inconsistency that could develop overtime among systems. Such data inconsistency would present difficulties for reporting and customer portfolio reconciliation.

Surprisingly, such a fundamental issue was not discussed and clarified more between the project team and the supplier. The project team seemed to have formed an expectation based on the supplier's initial product description and waited for the first product delivery on 30/04/2001 to understand the actual design. The tight timescale to meet an industry deadline meant that it would have been inconceivable for the project to switch the supplier at that stage. There was little chance for the project team to force AES to re-design the product either. The contract between SubB and AES simply did not cater for such an unexpected situation. All the project team managed was a feeble protest in front of AES programme manager toward the end of the project. The issue was a sensitive one that could lead SLC to be accused of negligence and was therefore not officially documented.

5.9.3 Data Integration

After GMRM was installed onto the client's system platform and before it could operate for production, a data migration exercise, also called data population, had to be performed to transfer relevant data from UMRS, SPMS and Transco to GMRM' database. This exercise went through a carefully managed process of requirement specification, script development, testing and execution (AES, 8, 9; SLC, 10). However, the data migration was problematic. Of the total target 469,034 records to be imported from Transco, 18,855, or 4% could not be imported for one reason or another (SLC, 12). The task of reconciling these records was left to the client to deal with separately after the project.

The new system had to operate with not only the legacy data, but also daily new data for production purpose. Of the six main business processes, customer gain and customer loss data could be processed successfully after the system went live on 23/07/2001. The processing of daily meter reads, however, could not be accomplished with the new system (SLC, 13, 14). Serious problems were reported including system "hanging" during loading meter reads, poor system performance and system had to be shutdown and restarted to progress data loading etc (SLC, 15). The hope of using GMRM to process meter reads for production was eventually abandoned. Temporary scripts were written by SLC to "convert" the reads into the right data format, which were then sent to Transco for gas usage reconciliation and also sent to SubB's billing system according to the established procedures (SLC, 14, 16).

Processing meter reads was a basic function expected of GMRM. In the initial acceptance testing, the basic meter reading process (Scenario 5 in AES, 18) actually passed the test (SLC, 7). In the "Known Omissions and Faults" document issued by AES on 23/05/2001 to accompany Release 1.01.00, AES claimed that all functions related to processing meter reads were included apart from "tolerance checking in validation" (AES, 10). This Release was subjected to the industry trial testing in June. SLC's test report stated that, though 22 defects

were raised on many aspects of the application, the Release *passed* the necessary tests for processing meter reads (SLC, 17). However, the industry trial test, like the acceptance tests, was conducted using carefully designed test data that amounted to a few dozen records (SLC, 17). These test data could not match the production data with "industrial strength"[20] (a term used by AES' Technical Architect). This industrial strength is not only related to data volume, but also to data variety that would demand a level of system robustness. Artificial test data give rise to artificial test results that could not be used to extrapolate on a system's full capability to deal with "industrial strength" data.

5.9.4 Product Acceptance Testing

According to Pfleeger (1998), a software application is expected to be tested first by the developers until "the system meets all requirements specified" (p. 360). Thereafter, the application is delivered to the client and is installed onto a system platform for the customer's acceptance testing. Writing from the developers' point of view, Pfleeger (1998) elaborates:

> "The purpose of acceptance testing is to enable the customers and users to determine if the system we built really meets their needs and expectations. Thus acceptance tests are written, conducted, and evaluated by the customers, with assistance from the developers only when the customer requests an answer to a technical question. Usually, those customers employees who were involved in requirements definition play a large part in acceptance testing because they understand what kind of system the customer intended to have built." (p. 360)

The above description, however, does not reflect how acceptance was carried out in GMRM project. Acceptance testing, like other formal testing activities, has to be guided by a set of detailed test scripts. The test scripts must specify exactly the inputs required, the actions to be performed and the expected system behaviours and outputs. Though customer employees involved in requirements definition know the business processes to be incorporated in the application, they do not know the technical implementation details before an application is delivered. Yet test script writing depends closely on the actual technical implementation in

[20] In the IT industry, industrial strength is the ability of a hardware or software product or a system to work capably and dependably in the operational world of business (TechTarget, 2001). Here the term "industrial strength" refers to volume and variety characteristics in operational data compared with artificially constructed test data.

terms of which buttons are to be pressed and which files are to be imported before expecting specific responses from the application. Even after a software application is delivered, customer employees must be trained by the supplier (the developers) to operate the system. Customer employees simply cannot be expected to write acceptance test scripts unless the software application is designed to be intuitive and for users who are expected to receive minimal training (e.g. web applications). For a business system like GMRM, complex data import and output operations together with the need to resolve application exceptions meant that the project team did not know what to expect before the application was delivered. As it turned out, AES provided acceptance testing scenarios and scripts (AES, 7; Section 5.7) in this project.

5.9.5 Software Product Documentation

Many documents related to the software product GMRM were issued from AES to the project team. The main ones were:

- GMRM User Guide (e.g. AES, 14);
- System Administrators Guide (e.g. AES, 15);
- Installation Instructions (e.g. AES, 16);
- GMRM Data Dictionary (e.g. AES, 17);
- GMRM Acceptance Test Scenarios (e.g. AES, 11, 12);
- Known faults and omission for a particular release (e.g. AES, 19).

Not all documents were issued with each release and not all documents were issued as complete. For example, the User Guide for Release 1 was issued as an "incomplete draft" (SLC, 28). The System Administrators Guide was not delivered until Release 2 in late June. Even then it was issued as a draft.

Much required information was missing. For example, GMRM was designed to validate input data according to validation rules with adjustable parameters. This was to deal with the

client's concern on meter readings that might have been mistaken in some obvious way (e.g. readings from a wrong meter). The validation capability was supposed to be available in Release 2.00.00 delivered on 27/06/2001 (AES, 19). However, no detail was provided by AES with the release. Once asked, AES provided a list of "error codes". Not happy with the information provided, the issue was escalated to AES' programme manager. AES replied with the same file reformatted with the file name changed to "Meter Reading Validation Rules"! The project team did not receive the requested information even after sending a similar document of validation rules as an example (SLC, 29).

5.9.6 Risk Management

A spreadsheet called Quantitative Risk Analysis Matrix (QRAM, see Appendix C for a template) was used to assess the overall project risks and the potential cost implications in the beginning (SLC, 3). The matrix produced for the project documented seven major risks, two of which were related to AES. Each was given a probability of occurrence, cost of occurrence, cost impact, mitigation action, mitigation cost delta (if the mitigation cost had been budgeted for already, the delta would be zero), the residual risk level and impact *after* the mitigation. Due to the risk analysis, the project team recommended an additional contingency fund of £98K to cover the mitigation cost delta plus the residual cost impact (SLC, 2).

A related but separate risk register was used to record and manage the on-going risks (SLC, 4). Some risks were foreseen and managed successfully. For example, one risk entry was related to the acquisition of hardware necessary for building system platforms for testing and production. Proactive actions were taken to expedite the procurement process. In addition, extra funding was secured (SLC, 30) to rent necessary hard disks to build a contingent test platform. This was in place for the delivery of the first GMRM release on 30/04/2001 while the acquired hardware was not in place until the end of May (SLC, 31). Some risks were

identified but were not successfully managed. For example, the risk of late software delivery with high rate of defects was prominent on the risk register with "visible evidence" based "monitoring" and "inspecting" of GMRM product development and AES' test results as the mitigating actions (SLC, 4). However, these actions did not appear to be feasible. The suggestion to visit AES premise during the development did not take place for fear of interrupting the development and slow down the software delivery further. The risk turned into a serious issue at the point of product implementation (Section 5.8). Another observation is that throughout the project, many project issues were dealt with on an as-required basis, without necessary being recorded in the risk register beforehand. For example, according to the project manager, the fact that the Project Sponsor was replaced half way through the project presented one of the biggest problems. Exactly what could be done as a mitigating action was not clear even if this had been foreseen. As a result, the risk register as updated on 06/08/2001 showed no further entry after 05/06/2001.

5.9.7 Post-Implementation Review

SLC's "Project Quality Plan" included a task of "Post Implementation Review" (SLC, 5). The action was "owned" by the project manager. While reviews did take place regarding individual team member performance, there was no review regarding the project as a whole or the GMRM workstream in particular by October 2001 when the project was closed down. In a private conversation in November 2001, the project manager confirmed the need to have a comprehensive post-implementation review. But he ruled out having an independent review by someone else in the company. He explained that that would have been contrary to the open and trusting culture SLC had always advocated. The author is not aware of the planned review has ever taken place.

5.9.8 Project Success or Failure?

The project can be described as "challenged" according to Standish Group (1995). The

project was completed in a way but delivered GMRM late and over budget with "fewer features and functions than originally specified" (Section 3.3.1). If the Maude and Willis (1991) criterion (a project is a failure if it would have been more economic not to have run it -- see Section 3.3) is to be applied, the project might be considered as a failure. This is supported by the fact that SubB later decided to replace GMRM with a product from another supplier[21].

5.10 MPS Project Chronology with Events Relating to GMRM

24 January 2001: SLC presented a feasibility study highlighting options available with input from various suppliers;

02 February 2001: SLC was selected as the prime contractor (SLC, 6);

28 March 2001: AES confirmed its contract proposal, increased license fee from the initial £250K to £350K (AES, 2, 6; SLC, 9);

01 April 2001: The original deadline for MRA/PGT Separation, which was postponed to July 2001;

03 April 2001: Transco called a meeting regarding industry trial for new systems to be compliant with Proteus system. The trial is to take place in June 2001 and a go/no go decision to be made on 05/07/2001;

03 April 2001: The contract between SubB and AES became effective ahead of the contract being formally signed;

10 April 2001: Project Initiation Meeting participated by management from SubB, AES and MPS Project Manager;

30 April 2001: SLC proposal to be the prime contractor was formally approved by SubB (SLC, 2). AES issued Release 1 as promised together with 28 acceptance test scenarios (AES, 7);

04 May 2001: SLC issued a test report on GMRM Release 1 (SLC, 7);

---

[21] See Footnote #18.

04 June 2001: UK-wide industry trial test started;

27 June 2001: AES issued Release 2 but not as defined in the original proposal. Release did not contain SubB specific functions;

28 June 2001: UK-wide industry trial test finish date;

05 July 2001: The decision was made by Transco and the other gas suppliers to go ahead with the new system implementation date of 23/07/2001;

23 July 2001: Transco Proteus went live. GMRM implemented and system support activities undertaken by SLC project team. Temporary converters were written to process data that could not be handled by GMRM.

29 July 2001: SubB outlined conditions for releasing £25K payment;

17 August 2001: Planned SLC project closedown date;

14 September 2001: AES issued GMRM2.02.00 which was equivalent to Release 2 as originally defined, over three months late. The release was subject to on-going test and defect fixes without being implemented to the production platform;

26 October 2001: SLC officially closed down the project. GMRM support was handed over to a dedicated support team. AES continued to develop GMRM that had not passed the acceptance test defined by AES itself.

5.11 Internal References

Reference#	Reference Source	Comments
AES, 1	GMRS For UKOneD1.pdf (content dated 27/3/2001)	MRMS (NDM) Product Description
AES, 2	R21024 Licence Proposal for MRMS version 1.pdf (file dated 28/3/2001)	Proposal to UKOne for the Supply of MRMS
AES, 3	R21024 Licence Proposal for MRM version 1.1.pdf (content dated 09/4/2001)	Proposal to UKOne for the Supply of MRMS - v1.1
AES, 4	R21024 Licence Proposal for MRM version 1.2 (content dated 03/05/2001, hardcopy only)	Proposal to UKOne for the Supply of MRMS - v1.2
AES, 5	AES Project Report 13 June.txt (content dated 13/06/2001)	AES Progress Report
AES, 6	Re UKOne License Proposal.txt (content dated 28/03/2001)	AES Confirmed Proposal
AES, 7	GMRS_AC_TEST_V1_2.doc (content dated 30/04/2001)	28 Acceptance Test Scripts Suggested by AES
AES, 8	MigrationV2.0drafta.DOC (content dated 11/05/2001)	Data Migration Guide
AES, 9	MigrationV2.0draftb.DOC (content dated 31/05/2001)	Data Migration Guide Revised
AES, 10	Known omissions and faults in version 1_1 MRMV1.doc (content dated 23/05/2001)	Known Omissions and Faults for Release 1.01.00.
AES, 11	GMRM_AC_TEST_V1_3.doc (content dated 21/05/2001)	Revised Acceptance Test Scripts
AES, 12	GMRM_AC_TEST_V1_4.doc (content dated 24/06/2001)	Revised Acceptance Test Scripts

Reference#	Reference Source	Comments
AES, 13	GMRM_FAT.XLS (file dated 28/05/2001)	Acceptance Test Data Supplied by AES
AES, 14	GMRM.chm (file dated 22/05/2001)	GMRM User Guide
AES, 15	System Administrators Guide V1.0draftc.doc (file dated 24/06/2001)	GMRM System Administrator's Guide
AES, 16	GMRM_Release_Installation_Plan.doc (file dated 26/04/2001)	GMRM Installation Guide
AES, 17	gmrm_data_dictionary.doc (file dated 24/04/2001)	GMRM Data Dictionary
AES, 18	GMRM_AC_TEST_V1_2.doc (content dated 30/04/2001)	GMRM Acceptance Test Scenarios
AES, 19	Known omissions and faults in version 2 GMRMDV1.doc (file dated 26/06/2001)	Known Faults and Omissions List
AES, 20	ErrorCodes.doc (file dated 17/07/2001)	GMRM Error Codes
SLC, 1	Gas Trading Report v1.ppt (content dated 24/01/2001)	Feasibility Study carried out for SubB
SLC, 2	MRA_PGT SPLIT Project Proposal v1.03.doc (file dated 30/4/2001)	SLC Proposal to SubB for the role of Prime Contractor
SLC, 3	DU005 QRAM_1.0.xls (file dated 26/4/2001)	Quantitative Risk Assessment Matrix
SLC, 4	MRAPGT_RISKS_ISSUES_REGISTER.doc (file dated 06/08/2001)	Risk Register
SLC, 5	MRA_PGT_Split_PQP_v1.0.doc (file dated 30/08/2001)	Project Quality Plan
SLC, 6	54087MRA.doc (content dated 02/02/2001)	SLC Bid/No Bid Decision

Reference#	Reference Source	Comments
SLC, 7	R1.00.00_Test_Report_SLC.doc (content dated 04/05/2001)	SLC Test Report on GMR Release 1
SLC, 8	BT_Highlight_JDI-015_010809.doc (content dated 09/08/2001)	Project Highlight Report
SLC, 9	MRA_PGT Highlight Report_010330v1.0.doc (content dated 02/04/2001)	Project Highlight Report
SLC, 10	GMRS Data Population Solution Requirements v2.1.doc (content dated 08/06/2001)	GMRS Data Population Requirements
SLC, 11	AES_Production_Support_Meeting_010904_AM.doc (content dated 04/09/2001)	Meeting Minutes Between SLC and AES
SLC, 12	GMRS_Data_Pop_Transco MRS v1.0.doc (content dated 15/08/2001)	Data Population Report for the Transco Data
SLC, 13	Daily Status Report 010724 v1.0.doc (content dated 24/07/2001)	GMRS Production Support Status Report
SLC, 14	BT_Highlight_JDI_015_010726.doc (content dated 26/07/2001)	Project Highlight Report
SLC, 15	PIR_Report_010809.xls (content dated 09/08/2001)	GMRS Defects Report Summary
SLC, 16	GMRS_Status_Report_Notes_010913_AM.doc (content dated 13/09/2001)	GMRS Production Status Report
SLC, 17	Shipper Trials Test Report v1.0.doc (content dated 07/07/2001)	Industry Trial Test Report
SLC, 18	R1.00.00_Test_Report_Cal.doc (content dated 04/05/2001)	Acceptance Test Report for the First GMRS Release
SLC, 19	gaspropcomparisons.doc (file dated 25/01/2001)	Supplier Initial Proposal Comparisons

Reference#	Reference Source	Comments
SLC, 21	DU005_Resource Costs v05.02.xls (file dated 18/07/2001)	Project Cost Monitoring Sheet
SLC, 22	DU005_Resource Costs v05.05.xls (file dated 07/08/2001)	Project Cost Monitoring Sheet
SLC, 23	DU005_Resource Costs v05.06.xls (file dated 16/08/2001)	Project Cost Monitoring Sheet
SLC, 24	DU005_Resource Costs v05.07.xls (file dated 22/08/2001)	Project Cost Monitoring Sheet
SLC, 25	DU005_Resource Costs v06.00.xls (file dated 27/08/2001)	Project Cost Monitoring Sheet
SLC, 26	DU005_Resource Costs v07.00.xls (file dated 13/09/2001)	Project Cost Monitoring Sheet
SLC, 27	Lessons_Learned_FormAM.doc (file dated 09/08/2001)	Project Lessons Learned for an Internal Workstream
SLC, 28	PMT_Meeting_Notes_010504.doc (content dated 04/05/2001)	Project Management Team Meeting Minutes
SLC, 29	RE FW GMRM Data Validation Document GMDM Validate Meter Readings.msg (content dated 19/07/2001)	Email Message from SLC to AES
SLC, 30	A Brief Business Case for Renting disks.msg (content dated 17/04/2001)	Email Message from SLC to UKOne
SLC, 31	Wednesday 30th May - Status Report.msg (content dated 30/05/2001)	Internal Email Message within SLC
SubB, 1	AES Contract - RE definition of last payment .msg (content dated 25/07/2001)	Email Message about AES Last Payment

Chapter 6: Case 3 – Building a Low Voltage Connectivity Model

(July 2001 to March 2002)

6.1 Background

The electricity distribution network cascades down from a high voltage (HV) grid to low voltage (LV) networks, and finally leading to consumer points. The Great Britain is divided into 14 Public Electricity Supply (PES) areas. Each PES area's electricity distribution network is controlled by a single distribution business, usually as part of a utility company. Due to recent privatisation and subsequent mergers and take-overs, some utility companies now control more than one PES area (Robinson, 1996; EA, 2003). The management of the distribution network within each PES area, however, has continued to be run as a natural monopoly (Kay, 1994; Helm, 1994) subject to the regulation of Office of Electricity Regulation (Offer, 1997), later Office of Gas and Electricity Market (Ofgem).

A distribution network may be susceptible to disruptions due to many reasons, including ageing equipment, extreme weather or even wild life activities. A disruption may cause an outage, i.e. a failure to distribute electricity to one or more customers. Among many aspects of work management within a distribution company, outage management is essential. Typically, an Outage Management System (OMS) is used to monitor, record and report outages in the distribution network and to help co-ordinating recovery. In an effort to improve connectivity and reduce the level of outage in distribution networks, Ofgem embarked on the second phase of the so-called "Information and Incentive Project (IIP)" on 18/01/2001 with a deadline of April 2002 (Ofgem, 2001a, 2001b). Two main objectives for the initiative are (Ofgem, 2001b):

- To reinforce the incentives for the distribution companies to meet quality targets;

- To introduce more relative comparison so that all distribution companies have a continual incentive to focus on performance, rather than meeting a single target.

Being a subsidiary of UKOne, SubC is responsible for maintaining the distribution networks in two PES areas (one is UKOne's "native" PES area and the other acquired in 1995 through take-over). Like other distribution companies, SubC initiated its own IIP project in response to Ofgem's IIP instructions and guidelines (Ofgem, 2000b, 2001b, 2001c, 2001d, 2002). The project was further divided into a number of workstreams or subprojects for effective management.

This case study examines one key workstream in SubC's IIP project. The workstream, known as LV Connectivity (LVC) Subproject[22], focused on the introduction of a software tool for linking customer information to distribution networks to produce an LVC Model. The successful completion of LVC Model with a certain minimum level of accuracy would enable SubC to meet part of Ofgem's regulatory requirements and recover 50p per customer (SubC, 8). Failure was not an option – if any company failed to comply by the deadline of April 2002, Ofgem would take actions including imposing financial penalties for the breach of licence conditions as set out in the Utilities Act 2000. This would include a penalty of up to 10% of the regulated income, which amounted to £40m in SubC's case (SubC, 8). In addition to meeting the regulatory requirements, SubC would gain operational efficiency by more accurate customer information, better outage management, better informed investment decisions and enhanced restoration of supply (SubC, 8). This case study first introduces the client's requirements followed by the project organisation and the selection of the software tool supplier. Thereafter various project aspects of cost, time and product are recorded in separate sections. The case is completed with within-case analyses on a number of issues.

---

[22] The workstream is sometimes referred to as a subproject, sometimes as a project. Where there is no confusion, both terms will be used interchangeably in this Chapter.

6.2 SubC Requirements Regarding LVC

A Statement of Requirements for IIP was compiled based on Ofgem's instructions and guidelines (SubC, 9). SubC's IIP as a whole had many aspects, some of which were carried out by internal SubC personnel, some assigned to contractors already in place. This case study focuses on the LVC Subproject, which was to establish an LVC Model (SubC, 9) for the two PES areas under SubC's responsibility. The LVC Model is a complete representation of all items in a LV distribution network, from feeders (wires connecting a substation to the LV networks) to customers represented by Metering Point Administration Numbers (MPANs). Ofgem also required the resultant LVC Model to be dynamic to the HV voltage level. That meant that as switching occurs on the HV network (i.e. the configuration of the HV systems changes), LVC model should be able to track and update in real time the HV feeding arrangements for all customers (Ofgem, 2000a). These requirements were to be met by April 2002.

Ofgem's LVC requirements were translated into the following objectives for the LVC Subproject:

- To acquire a software tool for linking LV feeders to customer MPANs;
- To build the LVC model with the software tool using existing customer information data and map-based distribution network data;
- To establish processes for LVC model to be maintained on an on-going basis.

The objectives were clearly linked. The first objective was the means to realise the second and the third. The successful acquisition and implementation of an efficient tool was essential before the LVC Model could be built. On top of that, detailed business processes to keep LVC Model to be dynamically up to date could only be devised with a clear understanding of how the software tool worked.

6.3 Project Organisation and Resources

SubC, as an organisation, had been well experienced in running mission critical ISD projects. The majority of the operational systems already in place had been developed with project teams directly managed by internal SubC project managers instead of relying on prime contractors or external consultants. IIP was also managed by an internal project manager with a senior business manager as the project sponsor. The project was broken down into a few workstreams or subprojects. Each workstream was managed by a subproject manager. The subproject managers were also from within SubC with business domain knowledge. They were assisted, among others, by one IT Project Manager who in turn commanded IT orientated staff in each subproject. The project organisation is shown in Figure 6.1 (based on SubC, 14).

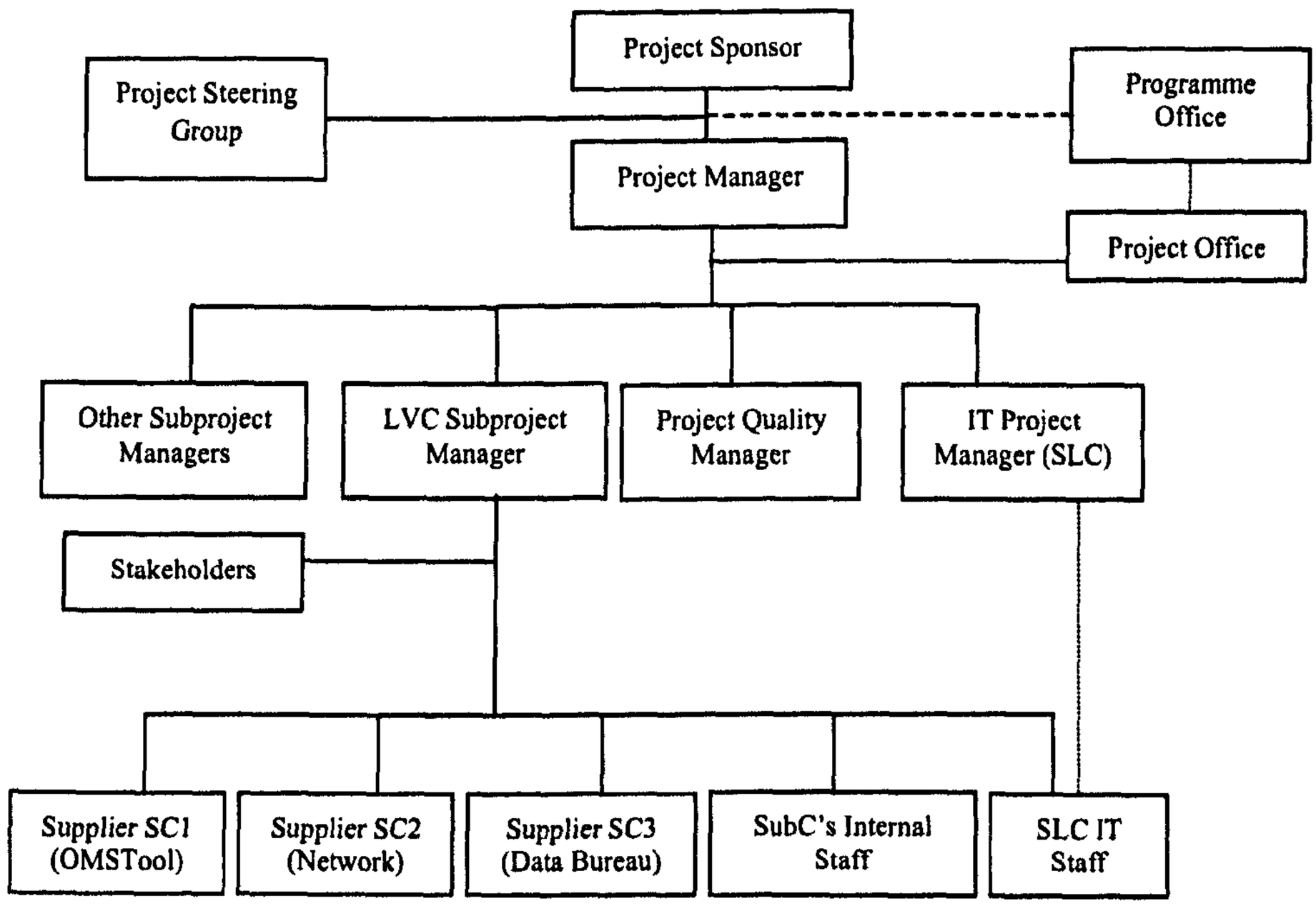


Figure 6.1: Indicative organisational chart for The IIP/LVC Project

The project organisation indicates that SubC was directly managing the overall project and various subprojects. Suppliers in the project were contracted and managed by client project managers to serve specific purposes. Supplier SC1 was selected to provide the software tool

OMSTool (See Section 6.4). Supplier SC2 was an existing supplier who provided the necessary IT network hardware and its maintenance services. Supplier SC3 was contracted to operate the "Data Capture Bureau", where the OMSTool was to be deployed and operated to construct the LVC Model on dedicated PC workstations. SLC's IT staff were largely responsible for integrating the OMSTool with client data and other systems by working closely with SubC staff and other suppliers.

6.4 Supplier Selection and Contracting

The LVC Subproject aimed to deliver a complex integrated information system consisting of the necessary system platforms, application software and relevant data repositories. The key was to acquire the application software that could assist building the required LVC Model accurately and efficiently. In addition, there was a need to integrate the system elements together as well as with the existing systems and infrastructure. This section describes the selection of the suppliers for the application software and the system integration.

6.4.1 Selecting the Application Software Supplier

An invitation to tender (ITT) document was sent to six potential software suppliers initially (SubC, 6). SubC required each supplier to complete a "Compliance Matrix". The Compliance Matrix listed SubC's requirements in eight major functional areas. Each area was further divided into a number of sub-headings varying from three to nine. For example, under the main heading of "Data Import Methodology", four different file formats were listed as separate requirements. The suppliers were invited to reveal the level of compliance of their software tool by stating whether it is compliant, partial compliant or not compliant at all. There were fifty-three subheadings in total. Two suppliers were short-listed for detailed assessment, one coded as SC1 and the other SC1A. Table 6.1 (SC1, 1) shows the level of compliance of OMSTool reported by SC1, which was the supplier eventually selected. For comparison, SC1A's compliance matrix is shown in Table 6.2 (SC1A, 1).

Table 6.1: Supplier SC1's perceived compliance summary

Requirement Headings	Count of Subheading Compliance Level				Total
	Compliant	Partial Compliant	Not Compliant	Not Clear	
Data Import Methodology	4	1	2	0	7
Core Functionality	7	2	0	0	9
Interface	7	1	0	0	8
Reporting	7	1	0	1	9
Infrastructure	6	0	0	0	6
Additional Functionality	7	0	0	0	7
Security	3	0	0	0	3
Output Structure	1	3	0	0	4
Total	42	8	2	1	53

Note: "Partial Compliant" includes "Compliant (with customisation)".

Table 6.2: Supplier SC1A's perceived compliance summary

Requirements	Count of Subheading Compliance Level				Total
	Compliant	Partial Compliant	Not Compliant	Not Clear	
Data Import Methodology	5	2	0	0	7
Core Functionality	9	0	0	0	9
Interface	8	0	0	0	8
Reporting	9	0	0	0	9
Infrastructure	6	0	0	0	6
Additional Functionality	7	0	0	0	7
Security	3	0	0	0	3
Output Structure	4	0	0	0	4
Total	51	2	0	0	53

Although there were subheadings in the Compliance Matrix showing "partial compliant" and "not compliant", the project team considered both suppliers' software tools to have met "the fundamental requirement" of associating MPANs to LV Feeders "fully" (SubC, 6). Therefore both software tools were considered to be "viable". In addition to the consideration of functional compliance, other non-functional aspects also were assessed. A summary document was produced, highlighting the strengths and weaknesses of the two options. An excerpt is given in Table 6.3 (based on SubC, 6, with some references to company names anonymised).

Table 6.3: Comparing the bids during supplier selection

	Supplier SC1	Supplier SC1A
Strengths	• Web Based Architecture - Matches UKOne's corporate IT strategy; • Currently used by 5 of 14 UK PES; • SC1 provide UK support; • Cheaper Product of two selected.	• The same product used successfully in SubC's sister company in US; • Experience in turnkey data delivery ; • Partnership with ICL, who know our business; • Offers 'One Corporate' solution.
Weaknesses	• Currently undergoing rapid sales growth, so support may be jeopardised; • Small Company; • No current experience of Turnkey data delivery.	• Product uses database that is not strategic to UKOne, therefore IT support will be more expensive; • Licensing is on user basis which may have costs issues for scaling up its use; • Supplier is US based, so support may be problematic due to time differences.

The project team's recommendation was to select SC1 on the ground of the lower price (20% lower compared with SC1A), more flexible architecture and UK support. Another crucial perceived advantage was that SC1's OMSTool product had already been deployed or were going to be deployed within five other UK utility companies, who were also undertaking IIP projects (SubC, 6). Though SC1's support capacity was identified as a weakness in the recommendation document, there was no evidence that this weakness was addressed. The lack of support capacity proved to be a serious handicap to the project when technical problems surfaced (Section 6.10).

It is worthwhile to note that the database technologies used in SC1 product were of open standard nature. This was marketed as strength by the supplier (SC1, 1). On the other hand, SC1A used proprietary database technology, which in addition was referred to as "not strategic" to UKOne's corporate strategy. There was, however, a favourable argument for SC1A's product due to SubC's sister company in US having "successfully" implemented it already. Overall, however, the decision was still in favour of SC1.

6.4.2 Contracting with SC1

Project team's recommendation for SC1 was duly approved. After a brief period of negotiation, SubC issued an award letter of contract to SC1 on 06/08/2001. The initial full contract document was not available for this study. An amended version of the contract letter dated 10/12/2001 gave details of a "Milestone Payment Schedule" (Table 6.4), which was based entirely on SC1's requested payment schedule dated 23/07/2001 (SC1, 1). Among the milestones, the key one was the system acceptance date of 14/09/2001.

Table 6.4:Milestone payment schedule between SubC and SC1

Milestone	Percentage Payment of the Total Contract Award Price
Following Contract Award	20%
Following delivering System Design Specification	10%
Following Site Delivery	35%
Following Site Acceptance (planned to be 14/09/2001)	25%
60 days after Acceptance	10%

There are two additional points in the contract amendment worth mentioning here (SubC, 7):

- A requirement of the supplier to have "Professional Indemnity Insurance" of £1,000,000 to be in place (SC1 claimed it to be);
- A contract sub-clause stating that "maximum loss recoverable by the client to be 100% of contract price or the value already paid".

However, there was no definition of conditions under which the "Professional Indemnity Insurance" would be claimed or the conditions under which the client would recover for any loss or the value paid.

6.5 Project Budget and Cost Control

Only budgetary details relevant to the LVC Subproject are discussed here. The initial budget expenditure for the LVC Subproject was set to be £1,943K (SubC, 29, 30). That figure was reduced to £1,400K (SubC, 10), mainly due to lower than expected software product cost. Percentages of various budget elements are given in Table 6.5 (based on SubC, 10).

Table 6.5: LVC Subproject budget breakdown

Item No.	Items	Percentage of Total Budget
1	SubC Internal Salaries	6.6%
2	SLC Cost for Integration	24.5%
3	Data Capture Bureau	36.4%
4	Data Capture Software OMSTool	17.1%
5	Software Tool Related Platform License Fees	5.8%
6	Reference Data Acquisition	3.7%
7	Data Capture Hardware	5.9%
Total		100%

In Table 6.5, the biggest expenditure for the project was budgeted for the Data Capture Bureau. This was due to the labour intensive nature of the capturing operation. The OMSTool (Item 4) together with the necessary system software licence fees (Item 5) cost 22.9%, less than the planned SLC system integration cost. This might be explained partly by the fact that OMSTool was a COTS software package, therefore the development cost was shared by a few clients. Partly it might be explained by the complex nature of system integration, which was carried out through phases of requirement analysis, solution design, development and implementation, demanding a high level of resources.

SC1's price for the OMSTool was fixed (Item 4 in Table 6.5), covering "computer software and associated data conversion of all UKOne's LV network data, to allow linking of customers to LV feeders, with associated outputs ..." (SubC, 11). The definition was comprehensive. SubC was able to manage the cost of OMSTool as fixed, giving little opportunity for SC1 to increase the price. Indeed, the project was successful in containing the project cost by March 2002, spending only £1,108K compared with the reduced budget of £1,400K (SubC, 32). On the other hand, system integration cost was based on a T&M contract with SLC. The costs of two other major items (Items 1 and 3 in Table 6.5) were also related to the length of project time and was initially budgeted assuming the project completion date was to be March 2002. As it turned out, the project had to be extended for an additional year, requiring £996K in addition to the initial budget of £1,400K (SubC, 31).

6.6 Product Quality Control and Product Acceptance

SC1's OMSTool was sold as a COTS product that should require only a limited amount of customisation. Not all quality procedures that OMSTool went through were disclosed. So far as the SubC's LVC project team was concerned, the following quality related activities were visible:

- SC1 presented a "Quality Manual" describing its overall quality system "designed to conform to International Standard BS EN ISO 9001:1994" (SC1, 3);
- A "Master Test Plan" was prepared up front (SC1, 4) dated 23/4/2001 together with the first version of the bidding document. This test plan "outlines the scope of testing, the way in which testing is to be conducted, who will test, and what resources are required when". However, this document remained as incomplete with its status left as "Draft" and a section on risks left with no details;
- Factory Acceptance Test (FAT) took place between 24/9/2001 and 28/9/2001 but the test report was not issued until 18/10/2001 (SC1, 5). The report contained no details on software defects other than stating the fact that 29 issues were recorded in the test;
- Site Acceptance Test (SAT) took place on SubC's system platform between 3/10/2001 and 5/10/2001. A number of issues were recorded and SC1 made an effort to provide fixes. SAT took place again between 11/10/2001 and 12/10/2001. A document was produced giving details of the issues (SC1, 6).

Due to unresolved critical issues after these quality control activities, the OMSTool was deemed not to be fully "accepted" by SubC. Therefore the milestone of product acceptance was left outstanding. Each subsequent patch release was tested with a subset of production data on a project test environment. The problem diagnosis and resolution was painfully slow.

6.7 Actual Project Progresses

The key milestone of Site Acceptance initially planned on 14/09/2001 was not achieved. Instead, there was a Factory Acceptance Test (FAT) on 24/09/2001 on SC1's premise. FAT was carried out on a system platform together with test data provided by the supplier. In other words, the supplier had a total control on the input. The project team merely witnessed the system to be operated. A project progress report described the FAT exercise as "a complete disaster in terms of converted data available for testing and core functionality" (SubC, 13). Even so, the project team decided to go ahead with the next stage of planned task, which was the Site Acceptance Test (SAT). SAT is usually carried out in order to allow a client to "accept" a software product. The first round of SAT was carried out from 03/10/2001 to 05/10/2001, producing 26 defects (SC1, 6), followed by a re-run of the same tests from 11/10/2001 to 12/10/2001. Due to the level of defects found, the client could not "accept" the software product. Even by April 2002, the acceptance milestone was not achieved (SubC, 2).

Problems with OMSTool impacted directly what SubC was able to achieve with the overall IIP project. A report from the Capture Bureau (SubC, 3) regarding April 2002's capture status showed that only 463,328 MPANs had been captured onto the LVC model out of a total of 3,368,725 (13.75%), a far cry from the initial plan of completion by April 2002. As a result, a separate effort by SubC had to be made to meet the Ofgem's minimal reporting requirements. According to a private conversation with a team member, the output of this separate effort was a "botched-up" solution, which could not be the basis of further maintenance. Therefore, the project had to continue with additional resources to achieve the original objectives[23].

[23] A conversation with a project team member who continued to be involved in the project indicated that the application was not stable with regular crashes even by November 2002.

6.8 Key Issues and Within-Case Analyses

OMSTool was sold to SubC as a COTS product. There were serious issues when it was implemented in SubC. Major issues were found in the following areas (SubC, 5, 12):

- Poor system performance;
- Problematic data conversion and integration;
- Poor documentation both for project communication and for OMSTool.

Each of these issues is discussed in a separate section below followed by observations on supplier response and risk management.

6.8.1 Poor System Performance

The performance gap between OMSTool as promised and the actual system installed was significant. SC1 claimed to have conducted performance test. In SC1's revised proposal dated 23/7/2001 (SC1, 2), a statement on the system performance was made (performance test is referred to as "load test" in this document):

> **"System Availability and Performance**
>
> The product has been load tested using the WinRunner[24] load test software. There were 500 concurrent users each creating transactions every 30 seconds on an AIX[25] Dual processor CPU and containing one gigabyte of RAM[26]. The load test produced negligible performance degradation compared to a single user.
>
> We understand that the number of users on this system is relatively small. The system is not responsible for processing high volumes of real time data. The main processor demands will be due to graphical based calculations on the server. These will arise from map operations or user queries. These are not likely to be very frequent given the small number of users."

It was true that there were only 20-30 users of the system in the Data Capture Bureau, far fewer than the 500 concurrent users tested allegedly. But poor system performance proved to be the single most serious problem impacting the project progress. Even after months of improvement effort, the performance difference between a single user and the required level

---

[24] WinRunner is a performance test tool supplied by Mercury Interactive Corporation (www.mecuryinteractive.com).
[25] AIX: the name of the Unix-based operating system marketed by IBM.
[26] RAM: random access memory.

of users in the Data Capture Bureau could not be described as "negligible" according to the actual test data (Table 6.6, based on SLC, 4). The response times with a single user were barely acceptable in that they failed to meet some of the minimum acceptance criteria. The performance with multiple users was totally unacceptable.

Table 6.6: Performance testing result: single user vs. multiple users on 17/01/2001

Action	Performance Acceptance Criteria (based on SubC, 1)	Performance Test Result (based on SLC, 4)	
	Maximum Response Time with Multiple User (above 30)	Single User	Multiple Users (between 21 to 27)
Login	2m	1m35s	3m28s
Pan	15s	13s	34s
Zoom	15s	20s	1m45s
Query	15s	20s	1m23s

The question was what caused the poor performance: was it the OMSTool or some other system elements? While system performance was clearly unacceptable according to testing, it was complex and slow to diagnose the root causes. The overall system was a combination of OMSTool from SC1, network supplied by SC2, PC Workstations configured by SLC and data provided by SubC itself. SC1 "claimed" that the fault was with UKOne's computer network. That was verified and rejected by SC2 (SubC, 33). Upon SC1's suggestions, SubC took other actions including re-configuring all operators' workstations (SubC, 34). No obvious improvement in system performance was achieved, demonstrating the uncertain process of diagnosing and resolving the underlying technical issues.

During many rounds of performance testing, some problems were discovered in the OMSTool software package. For example, toward the end of February, a fault was diagnosed that OMSTool displayed the same data twice (SLC, 5), thus taking roughly twice the time than necessary! Even when this problem was resolved, the performance remained unsatisfactory. In a review meeting dated 07/03/2002, SC1 admitted "the feel" that OMSTool performance was reaching a "plateau" and "there are no further obvious advances that can be made". SubC countered by holding back payment equivalent to 30% of the total contract value (SubC, 1).

6.8.2 Poor Interoperability between OMSTool and Data

Data was an important part of building the LVC model. SubC had all the required data available for the model in a number of other systems. The available data had to be transferred from the source systems, converted into the right format for OMSTool. There were two problems encountered in the process. One was data conversion and the other was data volume. The data conversion was declared to be "by far the greatest area for concern" in a project progress report in November 2001 (SubC, 15) when the Data Capture Bureau started operation. A few months later the converted data volume was high enough to expose the system performance problem (SubC, 16; Section 6.8.1).

While the poor system performance was not foreseen at all (Section 6.8.1), some warnings were to be found in the original proposal from SC1 if examined closely (SC1, 2). After re-assuring the client that "there should be no problem importing the data" based on their understandings, SC1 cautioned that there might be other problems with data formats and it was "a key area of uncertainty" (SC1, 2). Unfortunately, no specific measure was taken to manage this uncertainty.

6.8.3 Project Documentation

In Section 3.2.5, it is suggested that project documentation be grouped into project working documents and software/system documents. Neither type was satisfactory from SC1. The project issues list dated 02/11/2001 (SLC, 12) pointed out the problem with documentation in general, and specific problems with release notes (not available), test reports (not detailed enough and contained controversial claims) and user guides (minimal in content). Similar issues were raised on 28/01/2002 (SLC, 5). Bearing in mind of the initial target software acceptance date of 14/09/2001, it seemed to be the case that SC1 had given documentation a low priority.

The situation was recognised and accepted by the client to a certain degree. In one list of project issues (SLC, 5), the client clearly requested a lower priority to be given to resolving issues with documentation compared with issues on software functionality and performance. This could not be interpreted as a sign of encouragement for the supplier to neglect documentation. Rather, given the difficult choices between functionality and documentation, the priority had to be given to improving functionality.

6.8.4 SC1's Poor Response to Problems

SC1 was slow in responding to issues uncovered regarding OMSTool. The above problems of poor performance, data conversion and integration, and documentation were featured consistently from the initial round of testing (SLC, 12, 5). In a project summary document dated 09/04/2002 (SubC, 2), LVC Subproject Manager commented that the "general feeling is that the support from the Supplier has been under-resourced and the progress continues on an ad-hoc basis".

The project team was aware of the potential lack of support capacity at the supplier selection stage but no obvious effort was made to address the risk (Section 6.4.1). Two separate issues seemed relevant. One is that SC1 was a small company at the time of the project and had a limited resource pool. SC1 only had less than 50 employees in total yet boasted presence in the US, UK and Europe. The available resources were inevitably stretched to serve different customers in different geographical areas. Related to the above was how SC1 allocated its limited resources when there were conflicting demands. While LVC Project was still in progress and the software acceptance was yet to be achieved, the supplier won a new contract in the US. Should the supplier concentrate the resources to resolve the issues to achieve site acceptance or should the resources be diverted to the new project? The belief in the LVC project team was that SC1 did divert the resources away from SubC's contract. The belief was

corroborated by phone calls between LVC project team and the SC1's key staff travelling in the US. This was the case despite that SubC was holding back 30% of contract payment.

The practice of "co-locating" developers from the supplier with the client team might be suggested as a potential solution to secure undivided attention from the supplier. Unfortunately, this practice could not have worked in this project. According to a private conversation with a visiting technical staff from SC1, the supplier had a strict policy on controlling the source codes. Only the compiled code was allowed out of the company's premise due to the consideration of intellectual property rights. Therefore, any developer located on the client's premise would not be able to carry out code-level diagnosis and would have to be supported by other developers at the supplier's own office, which would not have been a feasible arrangement.

6.8.5 Risk Management

Risk management was emphasised from the early stage of the project. A guideline document drafted by the programme manager defined a risk as "an event, internal or external, that may cause the project to fail to meet its objectives" (SubC, 14). The document clearly specified that risk management was part of each subproject manager's responsibilities, alongside project planning and budgeting etc (SubC, 14). It was also mandatory to review risks as part of the weekly subproject meetings and monthly "stakeholder" meeting (SubC, 14). Many risks were identified. In a document dated 30/07/2001, 48 risks were identified for the LVC subproject alone (SubC, 28). The risks appeared to be actively tracked both at the subproject level and the programme level. Yet the options appeared to be limited to overcome the risks effectively.

Table 6.7 presents the risk tracking information from July 2001 to May 2002 regarding one of the main risks facing the project. The risk was referred to as the "Availability of Toolset to

Link Customers to Required Accuracy" (SubC, 17). The "Toolset" referred to the application software that was to be acquired from the external supplier. When the risk was first registered, the supplier selection decision had not yet been made. Following the supplier decision in July 2001, the confidence was high and the "residual risk" (the risk after taking the specified actions) was considered to be low for three months. In November 2001, a cautious but positive note was registered, stating "[s]oftware improvements ongoing". After January 2002 onwards, the residual risk level turned high and remained high despite the efforts made by the project team and the supplier.

Table 6.7: Risk Assessment on the Software Availability from July 2001 to May 2002

Month Updated	Risk Flag	Impact	Actions Taken	Target Date	Residual Risk	References
Jul-01	◁ ▷	High	Decision on Product to be made 9/7/2001	Sep-01	Medium	SubC, 17
Aug-01	▽	High	SC1 installing software on site. Satisfactory progress in accordance with IT Project Plan.	Nov-01	Low	SubC, 18
Sep-01	▽	High	SC1 installing software on site. Satisfactory progress in accordance with IT Project Plan.	Nov-01	Low	SubC, 19
Oct-01	▽	High	SC1 installing software on site. Satisfactory progress in accordance with IT Project Plan.	Nov-01	Low	SubC, 20
Nov-01	◁ ▷	High	SC1 installed software on site. Software improvements ongoing.	Nov-01	Low	SubC, 21
Dec-01	△	High	SC1 installed software on site. Software improvements ongoing.	Jan-02	Medium	SubC, 22
Jan-02	◁ ▷	High	SC1 installed software on site. Software improvements ongoing. Performance not at optimum level.	Feb-02	Medium	SubC, 23
Feb-02	△	High	SC1 installed software on site. Software improvements ongoing. Performance not at optimum level.	Mar-02	High	SubC, 24
Mar-02	△	High	SC1 installed software on site. Software improvements ongoing. Performance not at optimum level. Performance close to acceptable level (To be agreed with SC1).	Mar-02	High	SubC, 25
Apr-02	△	High	SC1 installed software on site. Software improvements ongoing. Performance not at optimum level. Performance close to acceptable level (To be agreed with SC1).	Mar-02 (sic)	High	SubC, 26
May-02	△	High	Site acceptance criteria not yet met. Tool performance has degraded. SC1 undertaking site investigations. Meeting on 8/5/2002 to discuss degradation and options to rectify.	May-02	High	SubC, 27

Note: In the column "Risk Flag", △ means "Risk increasing since last update", ◁ ▷ means "Risk unchanged since last update" and ▽ means "Risk reducing since last update".

Why did the project risk management fail to prevent this particular risk from impacting the project so adversely? Various explanations are possible. An inspection of Table 6.7 might suggest that the project team was overly confident of its supplier selection so that there was no viable alternative planned. As late as March 2002, the project team was still confident by stating that the software performance was "close to acceptable level", only to be disappointed by the performance degradation later. Due to the lack of any feasible option, the target date for resolving the risk was simply being pushed out reacting passively to events at the time.

6.8.6 Project Success or Failure?

The LVC Project did not produce an LVC Model within the planned timescale. By April 2002, the project processed under 14% of the target data volume (Section 6.7). Compared to the original project plan, the project could not be considered as a success. However, by the original project deadline of April 2002, the project was not cancelled. By Sauer (1993)'s narrow definition of failure, the project could not be considered as a failure either, since there was enough support to continue its operation. The successful aspects of the project were not just limited to its continued survival after the project deadline. The successful establishment of the Capture Bureau, a level of completion of the LVC model and the successful management of OMSTool's cost to be fixed can all be counted as successes in their own ways. It is perhaps fitting to describe the project as another "challenged" one according to Standish Group (1995).

6.9 Project Chronology

24 January 2001: High-level requirements documented within SubC;

Early April 2001: SubC issued Invitation to Tender (ITT) to potential suppliers;

Late April 2001: Responses to ITT received from suppliers;

May-June 2001: Evaluating suppliers' responses to ITT;

09 July 2001: Project Steering Group Meeting approved Project Team's recommendation on selecting SC1;

12 July 2001: Post Tender Negotiation meeting with SC1;

19 July 2001: A technical visit to SC1 to progress negotiation towards a contract;

26 July 2001: A meeting with SC1a was held to give an update on the tender process and the reasons for recommending the other supplier; (The above based on SubC;

06 August 2001: SubC issued an award letter of contract to SC1;

14 September 2001: Planned "Site Acceptance" Date;

24 September 2001: Factory Acceptance Test (FAT) started on SC1's premise;

28 September 2001: FAT finished, SubC internal report highlighted the FAT as a "disaster", but decided to proceed to Site Acceptance Test (SAT);

03 October 2001: First SAT on the client premise started;

05 October 2001: First SAT finished;

11 October 2001: Second round of SAT started;

12 October 2001: Second round of SAT finished;

18 October 2001: SC1 submitted reports on FAT and SAT;

10 December 2001: An amended letter of contract to SC1 outlining an agreed "Milestone Payment Schedule";

28 January 2002: OMSTool performance was cited as the a major issue;

18 February 2002: OMSTool performance was still the number one problem;

07 March 2002: SC1 stated "the feel" that OMSTool performance was reaching a "plateau" and "no further obvious advances that can be made";

05 April 2002: SubC reported that only 13.75% of the LVC model was complete, compared with the original plan of completion by then;

09 April 2002: LVC Project Manager reported that "Site Acceptance milestone has not been met and thus payment withheld".

6.10 Internal References

Reference#	Reference Source	Comments
SubC, 1	Acceptance Criteria and Payment Plan.doc (06/03/2002)	Client Suggested Acceptance Criteria and Payment Plan
SubC, 2	PM Handover Report - LV Connectivity 9 April 02.doc (09/04/2002)	Sub-Project Manager's Leaving Report
SubC, 3	Bureau Report WE 050402.doc (dated 05/04/2002)	Weekly Operational Report
SubC, 4	IIP Monthly Statistics May 02.doc (dated 05/06/2002)	Monthly Operational Statistics
SubC, 5	Main Issues with Software.doc (dated 28/01/2002)	Application Software Issues List
SubC, 6	LV Data Capture Software SC1 summary v1_1.doc (dated 06/07/2001)	Application Software Evaluation Report
SubC, 7	Amended_Contract_Award_Letter dec 01.doc (content dated 10/12/2001)	Client Contract Amendment
SubC, 8	Network Article Oct 01R.doc (file dated 04/10/2001)	Submitted Project Profile in an Internal Publication
SubC, 9	High Level Requirements as at Dec 2000.doc (file dated 24/01/2001)	High Level Project Requirements
SubC, 10	LV Conn Forecast NOV 01.xls (file dated 12/11/2001)	Budget Forecast
SubC, 11	Procurement Detail for IIP SC1 Systems 260701.doc (content dated 26/7/2001)	Procurement Details for Supplier SC1

Reference#	Reference Source	Comments
SubC, 12	Outstanding Issues prior to Site Acceptance v2.doc (file dated 02/11/2001)	Outstanding Issues with OMSTool as of 02/11/2001
SubC, 13	Update FOR 231001.doc (content dated 10/10/2001)	Project Progress Report
SubC, 14	IIP Project Managers Guide.doc (content dated November, 2000)	Internal Project Management Guide
SubC, 15	Update for 201101.doc (content dated 07/11/2001)	Project Progress Report
SubC, 16	IIP Project Board Progress Summary 18th February 2002.doc (content dated 18/02/2002)	Project Board Progress Report
SubC, 17	risk register by priority FOR JULY 2001.doc (file dated 20/08/2001)	Project Risk Register Dated July 2001
SubC, 18	RISK REGISTER AUG 2001.doc (file dated 12/09/2001)	Project Risk Register Dated August 2001
SubC, 19	RISK REGISTER SEPT 2001.doc (file dated 27/09/2001)	Project Risk Register Dated September 2001
SubC, 20	RISK REGISTER OCTOBER 2001.doc (file dated 12/10/2001)	Project Risk Register Dated October 2001
SubC, 21	RISK REGISTER NOVEMBER 2001.doc (file dated 15/11/2001)	Project Risk Register Dated November 2001
SubC, 22	RISK REGISTER DECEMBER 2001.doc (file dated 24/12/2001)	Project Risk Register Dated December 2001
SubC, 23	RISK REGISTER JANUARY 2002.doc (file dated 18/01/2001)	Project Risk Register Dated January 2002

Reference#	Reference Source	Comments
SubC, 24	RISK REGISTER 18th February 2002.doc (content dated 18/02/2002)	Project Risk Register Dated February 2002
SubC, 25	RISK REGISTER 19th March 2002.doc (content dated 19/03/2002)	Project Risk Register Dated March 2002
SubC, 26	RISK REGISTER 16th April 2002.doc (content dated 16/04/2002)	Project Risk Register Dated April 2002
SubC, 27	RISK REGISTER 14th May 2002.doc (content dated 14/05/2002)	Project Risk Register Dated May 2002
SubC, 28	LV ConnectivityRI 300701.doc (content dated 30/07/2001)	Risk Register for the LVC Workstream
SubC, 29	PROJECT FORECAST ALL PROJECTS TO MARCH 2002 AS AT JUNE 01.xls	Budget Forecast as of June 2001
SubC, 30	F100 Estimate Sheet LV.xls (file dated 06/06/2001)	Budget Request for April 2001- March 2002
SubC, 31	LV Conn F100 2002 .xls (filed dated 14/05/2002)	Additional Budget Request for April 2002- March 2003
SubC, 32	COSTING SUMMARIES FOR IIP PROJECTS MARCH 2002 ACTUALS.xls (file dated 12/04/2002)	Actual Project Expenditure for April 2001 to March 2002
SubC, 33	Minutes - Tech 17012002.doc (content dated 17/01/2002)	Project Progress Meeting Minutes
SubC, 34	Minutes - Tech 22022002.doc (content dated 22/02/2002)	Project Progress Meeting Minutes
SC1, 1	SubC LV Connectivity 1475 SC1.doc (content dated 23/4/2001)	Project Proposal from Supplier SC1

Reference#	Reference Source	Comments
SC1, 2	SubC LV Connectivity 1475 July 2001 v2.doc (content dated 23/7/2001)	Project Proposal from Supplier SC2
SC1, 3	SC1 Iso_man.doc (file dated 25/6/2001)	Quality Manual from Supplier SC1
SC1, 4	SC1 mastertestplan.doc (content dated 23/4/2001)	Test Plan from Supplier SC1
SC1, 5	SC1_FAT_Report.doc (content dated 18/10/2001)	Factory Acceptance Test Report
SC1, 6	SC1_SAT review1.doc (content dated 18/10/2001)	Site Acceptance Test Report
SC1A, 1	SubC ITT response 20-Apr-01CNM1v 2.0.doc (file dated 7/6/2001)	SC1's response to Invitation to Tender
SLC, 1	SLC Invest and Incentphase2v2.doc (file dated 10/4/2001)	SLC Internal Bid Review
SLC, 2	SLC IIPprop_v1.3.doc (content dated 28/6/2001)	SLC Project Proposal to SubC
SLC, 3	Update FOR 231001.doc (content dated 23/10/2001)	SLC Project Progress Report
SLC, 4	PerfReport170202_GLE.doc (content dated 17/01/2001)	OMSTool Performance Test Report
SLC, 5	PerfReport20020227_GLE.doc (content dated 27/02/2001)	OMSTool Performance Test Report

Chapter 7: Cross-Case Analysis Part 1: Describing the Pattern of ISD Project Challenges

7.1 Introduction

This chapter forms Part 1 of two-part cross-case analyses. The objective of this chapter is to outline a pattern of ISD project challenges while the following chapter (Chapter 8) attempts to explain the outlined pattern. This chapter is mainly based on the case studies documented in the previous three chapters. Where appropriate, however, published cases are quoted to shed further light on the issues under discussion (see Sections 2.4 and 2.6 for discussions of the research methodology for the cross-case analysis). The focus is on the client-supplier commercial relationship, supplier's product development, product evaluation and project risk management. Discussions are presented from the client's viewpoint unless made clear otherwise. Section 7.2 gives a brief overview of the three cases in Chapters 4, 5 and 6. Section 7.3 discusses the commercial aspects of application software supplier management including issues of supplier selection and contracting and client-supplier relationship. Section 7.4 examines the issues related to product development including requirements management, supplier capacity and documentation. The difficulties with software product evaluation are discussed in Section 7.5 and risk management in Section 7.6. A general pattern of ISD project challenges is suggested in Section 7.7. The chapter ends with a summary in Section 7.8.

7.2 Overview of the Three Participant Case Studies

Table 7.1 provides a comparative overview of the three cases described in the previous three chapters. Case 1 was initiated due to a vision from the client's board of management while the other two were undertaken due to regulatory requirements. Clients were closely involved in all projects. In Case 3 the project was directly managed by the client. The author's employer SLC was involved in these projects assuming different roles. For Case 1 and Case 2, SLC was

the prime contractor managing all suppliers on behalf of the client organisations. In Case 3, SLC sent IT resources to the project and managed only the technical part of the project. Suppliers' roles were varied and contrasting. Suppliers in Case 1 designed and developed software system elements with source codes entirely open for review. The contracts for all suppliers in Case 1 were of Time and Materials (T&M) type constrained by cost and schedule targets. The main software supplier AES in Case 2 was to design and develop a new application for a number of clients, one of those was SubB. AES retained the design documents as proprietary but released the software source codes to the client at the point of implementation. AES was to supply the software for a fixed price and to provide consultancy services on a T&M basis. In Case 3, the supplier SC1 supplied the software as a COTS product with a fixed price. The software was only released in an executable version with its design and source codes not available to the client.

Table 7.1: An overview of the three cases recorded in Chapters 4, 5 and 6

	Case 1 (WebShop, Chapter 4)	Case 2 (GMRM, Chapter 5)	Case 3 (LVC Model, Chapter 6)
Reason for undertaking the project	To realise a vision from client's board of management	To respond to regulatory requirements with a tight deadline	To respond to regulatory requirements with a tight deadline
Client's Role	Sponsoring the project and providing business resources to support the project	Sponsoring the project and providing business resources to support the project	Sponsoring the project and taking the overall project management responsibilities
SLC's Role	Prime Contractor on a T&M basis	Prime Contractor on a T&M basis	Providing resources to the project managed by the client directly on a T&M basis
Software Suppliers' Roles	Various suppliers to custom-build various software artifacts into a coherent web application	The main application supplier AES to design and develop a new application for a number of clients	The main software supplier SC1 to supply and implement a COTS application with commitment to convert legacy data.
Software Suppliers' Contracts	All suppliers on T&M based contracts	Fixed price contract for the software product GMRM T&M contract for AES consultancy services	Fixed price contract for the software product OMSTool T&M contract for SC1 consultancy services
Types of Software Products	CBSW	COTS (known to be in development at the time of contract)	COTS

The following discussions are based on the accounts of the three cases, supplemented by references to some of the case studies found in the literature.

7.3 Managing Supplier Selection and Contracting

Supplier selection and contracting might be regarded as the key to acquiring good software applications for ISD projects that rely on third party suppliers. However, evidence from the cases suggests that there are various issues in the process. These are discussed below.

7.3.1 Supplier Selection

When a client organisation decides to outsource the development of software products to an external company, it is important to select the "right" supplier. However, exactly who is the "right" supplier is not easy to determine. Three issues are discussed below: external interference of supplier selection, selection decision made under various project constraints and supplier selection based on claims rather than finished products.

External Interference of Supplier Selection

Ideally, a supplier should be selected to participate in an ISD project only if it can meet project objectives. In reality, however, there seem be considerations outside the immediate project objectives in the supplier selection process. Consider Case 1, the decision to involve MSS was related to SubA's desire to forge a marketing link with Microsoft and SLC's desire to gain skills and knowledge for Microsoft products (Section 4.7.2). The management decision directly interfered with the project operation, leading to the eventual selection of MSS as one of the main software development suppliers. When problems surfaced with MSS' lack of development capacity, the project manager did not have the authority to take decisive actions to replace MSS. An additional supplier had to be brought in at a significant extra cost. In the meantime, the relationship between the ISD project manager and MSS deteriorated, causing severe problems in project communication and project team morale.

A similar case of managerial interference in supplier selection is found in the RISP project

undertaken by Wessex Health Board (Collins, 1998). As a result of supplier lobbying and subsequent senior management interference on the client's side, the initial supplier decision by the project team was overturned and another supplier was contracted, almost in secret. The disastrous project result may not be attributed to the supplier selection process alone but the lack of its integrity did not help the project to achieve its objectives.

Supplier Selection under Severe Constraints

The selection process in Case 2 had a clear objective to produce an information system for meeting a regulatory requirement (Section 5.3). However, the selection was constrained by a number of factors. First of all, there was little time available to allow the required software product to be custom-built. This was due to the client organisation's delay in preparing for the project, an issue outside the scope of this study. Secondly, there were not many suppliers available for selection. It was considered to be a specialist area requiring a combination of industry knowledge and technical skills. Thirdly, in between the two suppliers available, the client had no practical choice other than selecting AES. This may be explained by the concept of switching cost. AES was the incumbent supplier for the core system supporting the client's existing gas operation with nearly a million customers. Selecting AES meant that the client's core systems could stay with peripheral adjustments while selecting the other supplier would have meant replacing the core systems at the same time (Section 5.3 and 5.9.1). Not only there would have been extra cost incurred, the risk and uncertainty involved in such a major system change meant that such a switch of supplier could not be contemplated. The concept of switching cost and its impact on supplier management is further discussed in Section 8.4.

Supplier Selection Based on Claims

Although there was no observed managerial interference in the supplier selection in Case 3, another common problem was observed in that software suppliers are selected based on suppliers' claims and promises rather than finished software systems (Section 6.4.1). Such a

selection decision is effectively based on many unwarranted assumptions. For example, some of the assumptions required supporting the selection of the supplier SC1 in Case 3 may be described as:

- That since core functions met the requirements already, "non-core" functions would not have presented many problems (Section 6.4.1);
- That since software performance was tested by the supplier and stated in the supplier's proposal document, it would perform accordingly after implementation (Section 6.8.1);
- That since there are other clients using the same supplier and the same product to meet the same regulatory requirement, it must be a sound choice (Section 6.4.1).

The decision-makers had a problem since there was no credible information available to indicate otherwise. It is not necessary that a supplier would deliberately lie on technical issues. Rather suppliers are inflicted with development uncertainty that may seriously impact on the development and delivery effort (see Section 8.7 on development uncertainty).

Of the above three issues, the problem of claim-based supplier selection is perhaps the most fundamental. At the supplier selection stage, suppliers are inclined to make necessary claims to win business. The lack of objective measurements also makes it easier for managerial intervention and allows an incumbent supplier to sway a suppler selection decision due to the client's high switching cost (Section 8.4).

7.3.2 Time Gap Between Supplier Selection and Contracting

There is normally a time gap between selecting a supplier and signing the contract with the supplier. During this period for Case 2, a surprise price increase of 40% was requested by the supplier AES, which was settled for 20% (Section 5.5.2). A remarkably similar price increase is reported in NAO (2003). In that case, the supplier ICL increased its price in between bidding and contracting. The client had to agree a 26% increase (from £146million to

£184million) after "intensive post-tender negotiations" (NAO, 2003, p. 14). Similar scenarios have been played out elsewhere. Lewis et al. (2000) record the following experience:

> "One of the more surprising problems occurred for a project that used a COTS product at one site on a pilot basis. The pilot implementation entailed a few hundred copies of the product; the full implementation was worldwide and would entail tens of thousands of copies. Once the decision was made to expand worldwide, the vendor raised the cost per copy. In the words of the project manager, 'We assumed that we would get a quantity discount. The price per copy is actually going up. The increase in price is so great that we are seriously considering starting over, this time writing the system ourselves from scratch.' A quantity discount had seemed so obvious that it was not included in the original negotiation."

Such price increases clearly create ill feelings between the client and supplier at the beginning of projects. AES behaviour during "contract negotiation" is not isolated. What is worth noting is the fact that suppliers get away with such price increases. One might think that, prior to signing a contract, the client has the right to change the supplier decision if circumstances change. In reality, however, there seem to be various hurdles in altering a supplier selection decision. First of all, the lack of time may not allow any delay in commissioning the supplier on software development (e.g. Case 2). Secondly, there may have been only a limited number of substitute suppliers available in a specific market. The initial decision might not have been based on price but on other considerations. While the selected supplier's price may have been increased, other factors may not have changed to warrant a change of supplier. Thirdly, to select a new supplier, the client may have to spend resources for further testing and evaluation. In addition to these material considerations, the decision-makers may have to overcome their pride to admit selecting the wrong supplier in the first place. All these factors may be explained by the concept of "switching cost", which is further explored in Section 8.4.

7.3.3 Supplier Contract Management

According to Bott et al. (2001):

> "Contracts set out the agreement between the parties: they set out the aims of the parties; provide for matters arising while the contract is running, ways of terminating the contract and the consequences of termination." (p. 117)

At the simplest level, a contract for designing, developing and supplying a software product should set out the agreed details regarding the product and other related matters. Supplier management then becomes a process of making sure that the supplier keeps to its contractual agreement. However, there are a number of observations indicating contract management is ineffective in ISD projects. First of all, the contracting process appears to be slow compared with the observed pace of projects. While lawyers may have to take time to study contractual clauses carefully, there is usually time pressure on project to be started and moved forward. This is most evident in Case 2 where the contract was not signed until half way through the project (Section 5.5.4). The phenomenon of engaging the supplier first and signing a contract later was also observed in Case 1 for MSS (Section 4.3.2). It is perhaps not surprising that this is the case since there is usually a substantial amount of money involved together with a need to finalise many details relating to complicated software products. The second observation is that, according to the available contracts, the agreement tends to be at a high level and incomplete, lacking necessary details to settle disagreements. Consider the clause for acceptance testing in Case 2, while the product acceptance depended on the acceptance testing, no detail of acceptance testing was specified in the contract (Section 5.5.3 and 5.5.4).

A general observation is that the contract types used in ISD projects, be it T&M contracts or fixed price ones, are not effective in managing suppliers' product delivery within cost and schedules. Consider Case 1, for example, MSS managed both to increase its price and reduce its deliverables with a T&M contract (Section 4.4). NQ managed to stay within the initial budget but only delivered a set of software artefacts without fully testing them due to the nature of its T&M contract (Section 4.6). With a fixed price contract, AES in Case 2 did not raise the price after the initial bid price increase. But the delivered software product lacked basic functionality and the available functions performed poorly (Section 5.8). Similarly, SC1 in Case 3 had a fixed price contract. The poor system performance, however, meant that the client had to extend the project for another year, costing an extra £996K (Section 6.5).

In short, current supplier contracts do not appear to fulfil the expectation of directing software suppliers' performance to achieve ISD project objectives. This is supported by other studies (e.g. Webster, 2000; Lichtenstein, 2002). The problems with supplier contracts are further analysed in Section 8.5.

7.3.4 Client-Supplier Relationship

At the early stages of an ISD project, the intention of working as "partners" is usually expressed by both the client and the supplier (e.g. Section 4.7.3). However, the partnership approach appears to be more a slogan than a project reality. Two observations might serve to illustrate this point. The first one is the potential conflict between the partnership approach and additional cost of maintaining the partnership. Consider the supplier NQ in Case 1. NQ did not complete the necessary activities as promised in its contract with the client with the designated budget limit. As a partner, NQ would be expected to fulfil its promises of delivering a quality product. However, NQ presented two options: either increase the budget or take the unfinished software artefacts. (Section 4.7.3).

The second observation is that the good will between a client and a supplier appears to dissipate over time. In Case 1, all suppliers were to be referred to as "partners" in the beginning (Section 4.7.2). In Case 2, the supplier pledged to work with the client as "partners". However, the client-supplier relationships were strained as the projects encountered difficulties. In Case 1, emotional arguments were frequent between the project manager and technical architect from MSS (Section 4.7.1). In Case 2, the supplier damaged the limited amount of trust in the beginning by requesting a surprise price increase. This was followed by poor quality of delivered software product. AES went so far as to develop the application with an altered architecture detrimental to the client's data integrity and on-going system support (Section 5.9.2). In the face of various project issues, it has been difficult to

reconcile the rhetoric of "partnership" with project reality. The concept of client-supplier partnership receives a critique in Section 8.2.

7.4 Managing Software Product Development

Within the context of an ISD project, a client expects a software product from the supplier so that the product can be integrated into the target information system. The software product is normally embodied in two types of deliverables (Section 3.2). One is software documentation; the other is the software executables and associated artefacts. The management of these deliverables are discussed below with a look at requirements management first.

7.4.1 Requirements Management

Poor requirements management has been identified in many studies as one of the root causes of ISD project challenges (e.g. Brooks, 1995; Bronzite, 2000). Two problems are often suggested: one is that of incomplete requirements and the other requirements changes. According to the author's observations, incomplete requirements do present a serious problem to some ISD projects. This is particular the case for Case 1 (Section 4.7.5). Consider the web page design, it was not quite possible to specify exactly how the web pages should have looked like, since the "look & feel" was very much one of the expected deliverables from the graphic designer NQ in the first place (Section 4.3.3). The problem of incomplete requirements has long been recognised. Brooks (1995) asserts that:

> "... it is really impossible for clients, even those working with software engineers, to specify completely, precisely and correctly the exact requirements of a modern software product before having built and tried some version of the product they are specifying." (p. 200)

However, requirements may be relatively complete, at least as the functional requirements are concerned. In Case 2, the client's requirements were partly imposed by the government regulations, partly decided by a common business process (acquiring and processing gas

meter readings), both were well understood by the supplier who was supposed to be an industry expert (Section 5.2). So far as the regulatory requirements were concerned, they were standardised and published as a set of dataflow interfaces. Considering Case 1 and Case 2 together, one question might be raised: exactly what requirements should be in place before they can be considered as complete for software development? While the answer may not be clear for Case 1, it is relatively clear for Case 2. The business requirements in the standardised form are all what a client can specify for the software supplier. In other words, there is a limit of what a client can specify for building a complete software product. The boundary of client requirements is further explored in Section 8.3 to explain the problem of incomplete requirements and its possible management implications.

The second problem often associated with requirements management is that of requirement changes or more often referred to as "scope creeps" (Field, 1997; Vowler, 1999; Nash, 2000; Kador, 2001). The allegation is that as a project progresses, the client asks for more and more deliverables. However, such a "scope creep" has not been observed in any of the three case studies. In Case 1, there was a clear understanding of the project scope in the beginning. As the project progressed, the scope was radically reduced rather than increased (Section 4.7.5). Case 2 and Case 3 were different from Case 1 in that the client requirements in both cases had been stable for the project durations. This was due to the fact that both projects were to meet a set of published regulatory requirements. Neither AES in Case 2 nor SC1 in Case 3 used requirements changes as explanations for their poor quality products. Whether requirements are stable or reduced, there seems to be a problem of aligning product development with requirements. This is discussed next.

7.4.2 Requirements-Driven vs. Delivery-Led Development

The observations in the case studies suggest that a process through which requirements-driven development gives way to one that is delivery-led. Here "requirements-driven development"

refers to a process that produces a software product that meets with a client's requirements. In contrast, "delivery-led development" process produces a product that is in some way convenient for the supplier. Unfortunately, the considerations for requirements and delivery are not always compatible with each other. Consider the reduction of the three-tier architecture to two-tier in Case 1 as an example (Section 4.4.2). The reduction reduced "the reliability, scalability and availability" of the target software product but did simplify the product development effort. The architectural change allowed the supplier to expend less effort but it was not accompanied by a formal variation. In this case, the supplier was going to deliver less for the same cost (or more). In Case 2, the architecture for the software product was changed to be stand-alone rather than integrated with an existing system the client already had (Section 5.9.2). The change potentially affected the client's long-term business data integrity seriously and would cost extra for system support. The change suited the supplier in that the software product may be sold to new customers (Section 5.9.2).

How could suppliers have adopted such significant changes in contradiction to client requirements? Of the two different examples discussed above, two different scenarios might be discerned. The first scenario is that a supplier, by trial-and-error, discovers that the initial design would take far more effort than the anticipated level to develop. Due to the pressure of time and resources, the supplier decides on an easier course to proceed, sacrificing the initial design objectives and product quality. Case 1 is likely to be an instance of this kind. The fundamental problem here might be related to the software development uncertainty, which is further analysed in Section 8.7.

The second scenario is that a supplier has a hidden agenda. The agenda is implemented first of all by not disclosing fully the design intention at the contract bidding stage. It is difficult for a client to understand software product design in abstract terms, more so if a supplier intends to keep certain details undisclosed. Of course, when a software product is developed,

the design has to be disclosed together with the product. By then, however, it is usually too late for the client to switch the supplier. Two important issues are at play in this scenario. One is that software design is typically embedded in the software product and can only be fully understood as part of the product, at least from the client's viewpoint. This is the inseparability of software product design and development that is further examined in Section 8.6. The other issue is related to the client's switching cost which is discussed in Section 8.4.

The need for "requirements-driven" software product has been emphasised in the literature (Gilb, 1997; Mylopoulos, 1999). It is worthwhile to note that the concept of "requirements-driven development" is presented here in contrast to "delivery-led" development. However, Royce (1998) uses the term "requirements-driven approach" to describe the attempt "to provide a precise requirements definition and then to implement exactly those requirements". As discussed in Section 7.4.1, it has been recognised that it is generally not possible to specify completely and unambiguously all requirements. Therefore, such a "requirements-driven approach" as described by Royce is not recommended and is not intended by the author.

7.4.3 Supplier Capacity Management

Even when client requirements are clearly understood and documented, the software supplier faces the issue of estimating and deploying the necessary resources for design and development within the constraints of time and cost. This is the classical problem of resource capacity management. The problem can be subdivided into estimating the amount of work to be done, assessing the resources required, securing the necessary resources and making sure that the work progresses according to the estimate etc.

The evidence from the cases suggests that software development suppliers do not manage their development capacity accurately and adequately, thus leading to unfulfilled promises. There are two potential contributing problems. One is that technical staff provide overly

optimistic estimates (Wiegers, 1998). Software suppliers make commitments to their clients based on these estimates resulting in insufficient time for developing quality products. However, behind the apparent software developers' optimism and estimating immaturity may lay a more fundamental problem of software development uncertainty (Section 8.7).

The other problem is essentially a managerial one. When technical teams produce one version of estimate, the supplier's management team may modify the estimate in order to be successful in contract bidding. The adjustment typically involves the cutting of resources to be competitive. Such an adjustment took place in Case 2. According to a private conversation with AES' technical architect[27], the development team's initial estimate for developing GMRM was £450K. The actual approved budget for the team was £250K. Due to the lack of accurate estimating techniques, it would not be easy for a development team to engage the management to debate which version of the estimates is more realistic.

In addition to coping with lower than necessary estimates made initially, a supplier may reduce its resources commitment to a particular project if re-deploying the available resources elsewhere helps it to maximise its profit. This was observed in Case 3. While LVC Project was still in progress and the full software acceptance had not taken place, the supplier won another contract. SC1 diverted resources away from the LVC Project even though SubC was holding back some 30% of contract payment (Section 6.8.4). A contract with penalty clauses may help. However, penalty clauses may also cause difficulties. In addition to inflating suppliers' bid prices, severe penalty may reduce incentives for suppliers to complete work if suppliers have already received the maximum possible payments (Bott et al., 2001), which is opposite to what the penalty is intended to achieve in the first place.

---

[27] This conversation took place between the technical architect and the author in September 2001. By the end of September, the technical architect left AES.

7.4.4 Interoperability of Application Software and Data

A successful information system is made up of a number of system elements, including hardware, system software, application software, operational data, system interfaces together with appropriate system documentation (Section 3.2). The current practice is that, based on the participant case studies, software suppliers are only responsible to deliver the application software and the associated software documentation. The system integrator is responsible for integrating the application software with all other elements including the client legacy data. While a software product may pass testing with a small number of carefully designed test datasets, it may not be able to process client's production data at the system integration stage.

Consider Case 2 as an example. During the initial acceptance test and the subsequent industrial trial, the software application was able to process meter readings with a few carefully designed test records. However, it was not able to import operational data that amounts to thousands of records everyday (Section 5.9.3). The difference between the operational data and the test data was both in terms of volume and quality. Operational data has an "industrial strength" (see footnote #17 for a definition) that no carefully designed test data can ever be expected to match. In Case 3, data played a significant role. The client had the required data items in various existing systems (Section 6.8.2) but not necessarily in the right formats. The interoperability of the supplied software and data was therefore paramount to the success of building the new data model. Yet the supplied software was expected to operate with the client data for the first time after the acceptance testing (Section 6.7). The problem appears to be particularly acute involving COTS software, as in Case 2 and Case 3. It is not clear exactly why the integration of applications and the clients' data is not attempted by suppliers before delivering software applications. In some instances, data protection has been cited as a major issue stopping clients' legacy data to be issued to suppliers. In any case, there appears to be a misconception that application software can interoperate with clients' data so far as the software is tested with a limited set of test data.

In integrating the supplied software and the client's legacy data, the integration team is asked to perform an impossible task. On the one hand, it is not technically equipped to diagnose and fix problems directly with a software application from an external supplier. In addition, it does not have access to the necessary design documents and it usually does not have access to the source codes. On the other hand, it has cost and schedule targets to meet within the overall ISD project constraints. All the project team can do is to report observed symptoms. Yet it is difficult to tell whether it is the data or the application or any other system element that causes the symptoms. The long period of time required for integrating data with delivered software in both Case 2 and Case 3 attests to an uncertain and unpredictable process (Section 5.9.3 and Section 6.8.2). In both cases, non-interoperability between the software and data led to repeated efforts to patch up the software applications and resulted in project delays and non-acceptance of software by the target project deadlines (Section 5.8 and Section 6.7).

7.4.5 Software Documentation and Its Quality

Software documentation is required, according to Brooks (1995), to enable software to be used, believed and adapted. As already discussed in Section 3.2.5, documentation is an important part of quality software applications and effective information systems. However, documentation is one of the major failings in ISD projects. Both project working and software/system documents have been known to be incomplete, often out of date and sometimes misleading. Poor project working documents hinder communications between the supplier and the client. Poor system documents affect the systems' use and support and increase the clients' cost of maintenance.

In the case studies, software documents were often observed to be inadequate. In Case 1, MSS made an attempt to document the overall system's architecture. The first version had the word "draft" appropriately marked on it and it was never updated further (Section 4.6.4). At

the point of the system going live in Case 2, the software application's documents were incomplete and what was available could not be relied upon (Section 5.9.5). In Case 3, the testing reports for releases were so poor that it was difficult to know which problem was fixed and which one was still to be resolved (Section 6.8.3). In every project, the priority was given to development and delivering the software codes, leaving documentation to the last stage of development, producing rushed and incomplete documents.

Brooks (1975, 1995) recognises the challenges of documentation and proposed "self-documenting" approach. Self-documenting refers to the practice of incorporating documentation in source codes. This might be helpful to technical staff responsible for software maintenance. However, the approach does not help clients in ISD projects. There are a number of potential problems that are not addressed by self-documenting. First of all, not all documents are related to source codes. For example, an IS architecture is a high-level design document, not possible to be documented as part of source codes. Secondly, commentary in a source program may help developers but not the operators who are not conversant with reading codes. Thirdly, commercial suppliers often withhold source codes as proprietary and only release executable codes, rendering the inclusive documentation inaccessible to the client's support and maintenance team. There is perhaps no easy solution to the problem of poor documentation. Good documentation demands highly competent resources (Nayar, 2000), often in competition with software development. This thesis suggests treating software documentation as an integral part of the software product (Section 10.2.1).

7.5 Evaluating Supplier Product

Each of the ISD projects described in the three case studies had test teams dedicated to evaluating software product quality (Sections 4.6, 5.7 and 6.6). In particular, acceptance testing was usually envisaged and planned for in the beginning of the projects. Yet the resultant software products were of poor quality. Why was quality not effectively controlled?

The problem appears to be related to the process of specifying and enforcement of acceptance criteria. By its name, acceptance criteria might be regarded as a number of conditions by which a product (an application or a system or indeed any product) is evaluated for acceptance or rejection. However, there are a number of problems with the concept when applied to a software product. First of all, the client has difficulty in devising a set of definitive acceptance criteria before a system is built (Section 4.6.2). At the beginning of a project, the client does not know exactly how a new system functions. If the acceptance criteria must be produced, some elements will be necessarily arbitrary. For example, before a website is built, it is not possible to be sure how fast each web page will respond. If a criterion must be given, a figure of one second may be as valid as two seconds. Such criteria are not helpful. Boehm (2000a) reports a case where the system's response was requested to be less than one second. It took a separate study to establish that it would have cost $100million to implement, compared with a cost of $30million for a response time of four seconds. The initial criterion was made irrelevant due to the associated cost.

Even when a list of acceptance criteria is available and not disputed, enforcing them until a product meets all criteria may cause long delays and increase the cost of a project significantly. Defining the criteria is just the start. A process must be in place with people and tools to facilitate the interpretation and verification of the criteria. From the supplier's viewpoint, if a product has to pass a fixed set of acceptance criteria, the product cost and the delivery time may exceed their targets. The supplier is likely to pass on the increased cost to the client wherever possible. With a fixed price contract, the client may not have to pay for any product price increase directly. However, it may still have to bear extra costs for retaining project staff due to a supplier's delayed product delivery. An added problem for a client is that ISD project teams, be it an internal or a subcontracted one acting on behalf of the client, is just as anxious to finish the project as the supplier is. ISD project managers are

usually measured on project budget and completion time as well as the product quality. Project budget and the completion time are much more visible and measurable than product quality. As a result, product quality is likely to be sacrificed and acceptance criteria might be ignored, as happened in Case 1 (Section 4.6.2). Alternatively, externally imposed project deadline may force a software to be deployed whether the acceptance criteria are met or not (Section 5.8).

The above observations suggest that acceptance criteria are not easy to specify and enforce. The difficulties with specifying might be related to the lack of a complete full design before development (Section 8.6) and certainly before contracting a supplier. The difficulties to enforce the criteria may be traced to the absolute evaluation approach, which is further discussed in Section 8.8.

7.6 Project Risk Management

In line with widely available guidelines on project management (e.g. CCTA, 1998; PMI, 2000), all three projects emphasised risk management as a key part of project management (Sections 4.7.7, 5.9.6 and 6.8.5). The risk management processes and guidelines were followed to a certain degree. Some of the risks were foreseen and managed successfully. Some were foreseen but not successfully managed. The single most significant risk not successfully managed was the late delivery and poor quality of the required software product in each of the project. This risk was in fact identified, assessed, documented with "mitigating actions" planned in the beginning of each project (Table 7.2). The risks initially identified, as shown in Table 7.2 were accurate in describing the issues later encountered in each of the projects. However, the mitigating actions were inadequate in overcoming the issues, resulting in the risks manifesting themselves as serious issues. Consider Case 1 first, the mitigating actions amount to an attempt to a "complete contract". This is in fact not easily achieved with incomplete requirements (Section 7.4.1 and Section 8.3) and the ineffective contract types

used in the current ISD projects (Section 8.5). In Case 2, "monitoring" and "inspecting" of the supplier product development was not really possible due to the hidden information and hidden action problem inherent in the "principal-agent" relationship (Section 5.9.6; Section 8.2). The risk registers from Case 3 provide a detailed historical record of how a risk turned into a serious issue (Section 6.8.5). By the end of the project, the client was left discussing the issue and seeking options with the supplier – rather late for the project. In all cases, the key problem appears to be that the proposed actions did not provide any practical option. The underlying reasons for the failure of risk management in ISD projects and possible management options are further explored in Section 8.7.

Table 7.2: Comparison of the risks related to the required software products and the suggested mitigating actions in across three projects

Project	The Main Risk Related to The Software Products	Proposed Mitigating Actions	The Outcome of the Products
Case 1: WebShop (see Section 4.7.7)	Timely and cost effective delivery.	Clear supplier responsibilities identified and enshrined in contract. Supplier Management responsibilities clearly identified.	MSS/IFG delivered late and cost more.
Case 2: GMRM (see Section 5.9.6)	GMRM not meeting the delivery date and having "an unacceptably high incidence of errors".	Visible evidence based "monitoring" and "inspecting" of GMRM product development and AES' test results.	GMRM Release 2 was delivered late and did not function as promised.
Case 3: OMSTool (see Section 6.8.5)	Availability of Toolset to Link Customers to Required Accuracy.	Various actions to manage the supplier, including "discuss degradation and options to rectify".	OMSTool did not perform to allow the LVC Model to be built by the project deadline.

7.7 The Pattern of ISD Project Challenges

What has been described in the previous sections is largely based on the three case studies in the previous three chapters. To what extent do these observations represent a general picture of ISD project challenges? To answer this question, it is helpful to quote at length a "pattern" described by Yourdon (2002):

> "[...] an ambitious business organization decides that it needs an innovative new information system; it then contracts with an aggressive software development

firm to build the system. At the beginning, the atmosphere is one of enthusiasm and optimism; the customer promises to devote resources and energy throughout the project, the vendor promises to provide high-quality, 'professional' services, and both parties pledge to work in a 'partnership' mode. A contract is generated and duly signed [...].

After a period that ranges from a few months to a couple years, things begin to turn sour. Deadlines come and go; budgets are exceeded; amendments to the contract are hurriedly negotiated and signed; and the mood gradually turns grim and hostile. Sooner or later, one of the parties abandons all hope, and terminates the contract; in most cases, it's the client who pulls the plug, complaining that the project has taken twice as long as initially promised, and has cost twice as much as originally budgeted. And to add insult to injury, the system either doesn't work at all, or is full of bugs, or runs so slowly that only 3 users can access it concurrently, instead of the 300 that had been promised.

At that point, both parties bring in their respective lawyers. [...]'experts' like me are brought in to advise the lawyers, the judge and the jury. All of them want credible answers to some very basic questions: whose fault was it that the project failed? How did it happen? When should it have been evident that the project was in trouble? Why didn't someone, on either one side or the other, do something about it?

If you don't think such things can happen in the 'real world', let me assure you that they do; indeed, the cost of litigation now exceeds the cost of coding for most large software development organizations. Only a small percentage of such failures actually result in lawsuits; a much greater percentage of the failures result in both parties walking away from the mess, swearing never to speak to one another again."

Striking similarities might be discerned comparing what has been described in the previous sections with Yourdon's "pattern". First of all, each of the three projects started with optimistic expectations. Especially for Case 1, it started with a "vision" from the corporate board (Section 4.1) and a "partnership" approach was very much emphasised as a way to work between the contracting parties (Section 4.3). Even for Case 2 and Case 3, where projects were initiated in response to regulatory requirements, promises from suppliers made the project successes seemed inevitable (Section 5.5 and Section 6.4). Secondly, control on project schedules and costs seemed difficult, resulting late deliveries and budget overruns. This is despite adopting aggressive scope reduction or scope-freezing strategies (Section 4.4, Section 5.8). Thirdly, software products from the suppliers failed to meet clients' requirements and do not match initial supplier promises in terms of functions, features and quality (Sections 4.6, 5.8 and 6.7). Though none of the three projects studied were officially

cancelled before the project closure, and the author is not aware of any lawsuit being filed, the clients in the case studies were certainly dissatisfied with the project outcomes. Similar patterns of ISD project challenges might be found in Webster (2000), Ward (2001) and other sources.

This study contributes to the description of the pattern by highlighting additional observations in the stages of supplier selection (Section 7.3.1) and product evaluation (Section 7.5). In the areas of the commercial management and product development, the case studies have highlighted considerable level of details (Sections 7.3 and 7.4). In addition, the failure of risk management has been described as part of the pattern contributing to ISD project challenges (Section 7.6). Of course, this study is not designed to apportion blame to any specific party in the case studies. Instead of asking questions like "whose fault was it that the project failed", the focus is on the root causes of the project challenges and possible remedial actions. Chapter 8 uses a number of existing concepts and theories to explain the pattern of challenges described above.

7.8 Summary

This chapter provides a survey of common ISD project challenges based on the cases recorded in the previous chapters and those published. The challenges have been examined mainly from the client's viewpoint. The focus has been on the process of client-supplier interactions. The management of commercial relationship with a supplier presents an ISD project team with significant challenges. Suppliers may be selected with external interferences weakening the authority of the project team. Supplier selection may be subjected to various constraints and worst of all, all suppliers have to be selected based on claims and promises in the current project management framework. Supplier contracting has been observed to be difficult due to client's high switching cost and also ineffective in directing suppliers to delivering quality products within specified cost and schedule targets. The client-

supplier partnership has been observed to be more of a slogan than a mechanism to resolve any real issue of cost or quality performance.

When it comes to product development, a number of challenges are present in ISD projects, ranging from requirements management, delivery-led development and the apparent difficulties for suppliers to keep to their cost and schedule estimates. When software products are delivered, the lack of interoperability between application software and client's legacy data causes further project delays and cost overruns, and sometimes threaten systems integrity altogether. Software documentation has also been observed to be of poor quality. The seemingly simple concept of acceptance criteria appears to be nearly impossible to implement in ISD projects since they are difficult to define and enforce. Risk management has not helped ISD projects to meet their objectives even when some of the risks are correctly identified and documented. The overall pattern of ISD project challenges encountered and described matches and enriches the descriptions already available in the literature.

Chapter 8: Cross-Case Analysis Part 2: Explaining the Pattern of ISD Project Challenges

8.1 Introduction

In the chapters 4 to 6, it has been confirmed that there are indeed significant managerial and technical challenges associated with ISD projects. In Chapter 7, it has been further confirmed that there is a pattern of ISD projects challenges. What are the root causes behind this pattern? In the course of the last three decades of research and practice, many hypotheses have been put forward as explanations. Technical limitations (hardware memory limit, programming language design etc), lack of training for practitioners, chaotic processes, poor project management and inadequate client organisations have all been blamed (e.g. Brooks, 1975; Humphrey, 1989; Flowers, 1996; Collins, 1998; Bronzite, 2000; Finkelstein, 2001). Corresponding solutions in the shape of new programming paradigms (object-orientated design and programming), training and various ISDMs and best practices have been prescribed (Section 3.5 and 3.6). However, that these solutions have failed to overcome the pattern of ISD project challenges suggests that the current explanations, likely as they may be, are not "lovely" (Section 3.7).

In this chapter, the author attempts to provide a potential best explanation for the pattern of ISD project challenges following the IBE reasoning strategy (Section 2.6.3). In line with the "study proposition" (Section 2.4.2), the process of software supplier management is closely examined in this Chapter. The client-supplier relationship is critically reviewed first in Section 8.2. Thereafter, a number of root causes are suggested based on concepts from agency theory, transaction cost theory and literature on product development and project risk management. Incorporating these discussions, the stages and milestones of a typical software

supplier selection, contracting and delivery process are described in Section 8.9 as the underlying traditional systems development framework (TSDF). This is followed a game theoretic model for TSDF with software suppliers making "non-credible" promises. Section 8.10 provides a critique of the popular iterative approach. Section 8.11 suggests a unified explanation to the observed pattern of ISD project challenges. Section 8.12 ends the chapter with a brief summary.

8.2 Client and Supplier Relationship

In an ISD project where a third party software supplier is contracted to design and develop application software, the client and the supplier enters into a close working relationship. The nature of such a relationship is important for understanding its mechanism and possible associated problems. This Section examines the current view of client-supplier partnership based on observations from the case studies. It is suggested that the relationship is better modelled as that of principal and agent.

8.2.1 Client-Supplier as Partners?

Client-supplier partnership, also referred to as supplier partnership, has been a popular term in the purchasing literature (e.g. Ellram, 1991; MacBeth and Ferguson, 1994; Brown et al., 1994; Smith and Tonks, 1995) and was much talked about in ISD projects like Case 1 (Section 4.3). Indeed, it is an approach advocated by many ISDMs (e.g. Royce, 1998; Highsmith, 2000). But what is client-supplier partnership? The concept seems to have originated in the manufacturing industry in Japan in 1970s (McGloin and Grant, 1997). It has since spread to the West and it "has established itself in various forms, significantly in the UK process engineering and construction industry" (Smith and Tonks, 1995). However, there are conflicting attitudes in the public sector regarding supplier partnership, with the European Union Directives making competitive tendering compulsory (Smith and Tonks, 1995; Parker and Hartley, 1997) and the UK Government promoting "partnering relationship" (OGC,

2002). Unfortunately, definitions for supplier partnership or "partnering relationship" have been at a high level and vague. For example, Smith and Tonks (1995) describe it as:

> "Partnering is a way of doing business which recognises that common goals exist which can be achieved through cooperation and open communications. However, it is more than a set of goals and procedures; it is a philosophy, a state of mind that demands commitment to respect, trust, cooperation and excellence for all stakeholders involved." (p. 119)

OGC (2002) simply states that "[p]artnering is where two, or more, organisations develop a close and, generally, long-term working relationship". The cooperative intention as defined at such a high level is difficult to fault and it seems that it should be universally applicable. The approach has been assigned many "advantages" compared with the competitive strategy according to Lamming and Cox (1995, quoted in Parker and Hartley, 1997).

However, Ackerman (1996) reckons that "partnership may be doomed to fail" as a result of his experiences in the logistics business. Parker and Hartley (1997) conclude that there is "no *a priori* case" in favour of either the partnership approach or the competition based supplier management approach, each "has advantages and disadvantages that will vary depending upon the precise circumstances" (p. 124). Cousins (2002) goes so far as to dismiss the partnership approach altogether, arguing that "[p]artnership relations do not exist" (p. 71). This Section aims to provide a critique of the partnership argument within the context of ISD projects.

The critique focuses on four aspects. The first is the basis of partnership, which is trust and goodwill according to MacBeth and Ferguson (1994). MacBeth and Ferguson elaborate different types of trust (contractual, competence and goodwill) and then emphasise that trust should be developed, "not blindly given without patterns of evidential behaviour" (p. 105). Therefore, it is reasonable to conclude that supplier partnership is only possible for repeat relationship between a client and a supplier. This has been the case for the manufacturing industry where a client typically purchases components repeatedly over a sustained period of

time, for which the concept of supplier partnership was conceived in the first place (MacBeth and Ferguson, 1994; McGloin and Grant, 1997). For an ISD project, the client and the supplier are engaged in a development process for a "unique" product (PMI, 2000) without necessarily "patterns of evidential behaviour" on which to develop the required level of trust, however desirable it might be. Though reputation has been regarded as a supplementary mechanism to constrain suppliers' behaviour, suppliers take a calculated approach to maintain their reputation rather than at all cost (Mailath and Samuelson, 2001). In addition, due to complexity and uncertainty in ISD projects, clients can often be blamed (e.g. Anthes, 1994; Johnson, 2000a; Buxbaum, 2001), allowing suppliers to defend their reputation in case of project difficulties. Thus the concept of supplier partnership, either based on client direct experience or indirect reputation mechanism, is not easily transplanted to ISD projects, or other similar situations where clients and suppliers are engaged in unique and risky ventures.

The second aspect is related to the sharing of profit and loss. Partnership in the traditional business sense refers to a form of organisation where members (partners) share profit and assume "joint unlimited liability" (Parkin et al., 1997, p. 203). The conditions for a partnership in this sense are therefore a): that the profit and liability should be clearly measurable and b): that the profit and liability should be shared between the partners. When these two conditions can be met client-supplier partnership might be possible. For example, it has been reported that EDS operates a partnership with the Inland Revenue of the UK Government on the basis of sharing cost savings (Hynes et al., 2000). Exactly how the cost savings have been measured may deserve further investigation. In general, however, the costs and benefits of an information system themselves are difficult to define (Bannister et al., 2001; Waller, 2002). This makes the sharing of the benefits difficult. Typically, a conventional procurement contract does not set out to share such benefits at all, measurable or not (Lacity and Hirchheim, 1995). Unlike a conventional business partnership where legal enforcement of the contractual obligations is possible, supplier partnership has remained a

philosophy and "a state of mind" (Smith and Tonks, 1995). This might explain the observation that "the partnership approach" has remained more as a slogan while actual client-supplier relations may deteriorate in the face of project difficulties (e.g. Section 7.3).

The third aspect is the single supplier relationship implied by the partnership approach, since more than one supplier for a particular good or service would mean a competitive strategy in the first place (Parker and Hartley, 1997). When managing a single source supplier, the client relies on a regulatory mechanism rather than a competitive one. Therefore partnership approach reduces the client-supplier relationship to that of the regulator and the regulated. It is generally considered in the economic literature that competition is superior to regulation (Beesley, 1997, quoted in Parker and Kilpatrick, 2002). As a result, "a goal of regulation is to promote competition" (Parker and Kilpatrick, 2002). To advocate managing suppliers via regulation rather than competition if and when there is a choice is contrary to the basic economic understanding about the two options. The fourth aspect is related to the third one in that single source approach is inherently anti-competitive. This is particularly problematic for public sector purchasing where there are strict legal obligations to pursue a competition-based procurement policy, though some confusion seems to exist in terms of actual implementation of the competition policy (Parker and Hartley, 1997).

Due to the above issues, the study of software supplier management below is not based on the assumption of client-supplier partnership. Instead, it is suggested that a client and a supplier form a principal-agent (PA) relationship informed by agency theory. The next Section looks at the PA relationship and how it might guide this study.

8.2.2 Principal-Agent Relationship

PA relationship, also called agency relationship, is common in economic transactions. The relationship arises when "the 'principal' is obliged to hire an 'agent' with specialized skills or

knowledge to perform the task in question" (Sappington, 1991). This description fits well the client-supplier relationship in an ISD project. In this case, the client would be the principal and the supplier the agent. The "task in question" would be designing, developing or customizing a software application for the client's organisation. The PA relationship is the focus of the agency theory, which is concerned with "how the principal can best motivate the agent to perform as the principal would prefer, taking into account the difficulties in monitoring the agent's activities" (Sappington, 1991). Arrow (1991) compares the agency theory with the "more standard economic theory:

> "The principal-agent theory is in the standard economic tradition. Both principal and agent are assumed to be making their decisions optimally in view of their constraints [...]. It offers insights used in the construction of contracts to guide and influence principal-agent relations in the real world; at the same time it represents an attempt to explain observed phenomena in the empirical economic world, particularly exchange relations that are not explained by more standard economic theory." (p. 38)

The agency theory is closely related to the transaction cost theory in that they share the basic assumptions of an economic agent's self-interest and bounded rationality (Williamson, 1985; Eisenhardt, 1989a). It is the combination of the two that have led to "serious contractual difficulties" (Williamson, 1985, p. 67). These assumptions are in stark contrast to the supplier partnership approach where supplier opportunism is not considered.

According to Eisenhardt (1989a), the focus of the agency theory is on "determining the optimal contract, behaviour versus outcome, between the principal and the agent" (p. 60). In other words, it examines the choice between a behaviour-oriented contract (e.g. salaries and hierarchical governance) and an outcome-based contract (e.g. commissions, stock option and market governance etc). In order to analyse the principal-agent contractual relationship, Eisenhardt (1989a) suggests examining a number possible variables. Relating to the study of ISD projects, the following issues are considered: information incompleteness, task programmability and outcome measurability, and outcome uncertainty. Other assumptions and variables may be considered for studying contracts in other circumstances. For example,

the removal of the assumption of self-interest or goal conflict may apply to situations involving "clan-oriented firm" or "selfless behaviour" (Eisenhardt, 1989a, p. 62), which are not discussed here.

Information Incompleteness

The basic assumption of bounded rationality in the agency theory implies incomplete information in the PA relationship. Incomplete information leads to the problem of hidden information, which refers to the misrepresentation of ability by the agent (Eisenhardt, 1989a). Arrow (1991) describes it as:

> "[...] the agent has made some observation that the principal has not made. The agent uses (and should use) this observation in making decisions; however, the principal cannot check whether the agent has used his or her information in the way that best serves the principal's interest." (p. 39)

Hidden information poses a problem for supplier selection in an ISD project. A supplier responds to a client's request for proposal (RFP) by giving a proposal with price, delivery and product quality information. The client does not have an easy way to verify if the information provided is more for winning the contract or for demonstrating the true capability to supply a superior product at the stated price. The problem of hidden information is considered further together with the concept of switching cost in Section 8.4.

While incomplete information makes it difficult for the client to select the best supplier, it also causes problems in terms of communicating the client's requirements to the supplier. Incomplete requirements have been cited often as an explanation for ISD project challenges (Standish Group, 1995; Keil et al., 2002). The issue of the principal's requirements does not appear to have been emphasised in the agency theory literature. However, the right level of requirements information is a prerequisite for the subsequent task completion. The issue of incomplete requirements appears to be particularly acute in ISD projects and is discussed in Section 8.3. Incomplete information at the contractual stage leads to a general problem of

incomplete contracts, which have been observed in the participant case studies (Section 7.3.3) and published literature (Webster, 2000). The problem is examined in Section 8.5.

Task Programmability and Outcome Measurability

According to Eisenhardt (1989a), task programmability is defined as the degree to which appropriate behaviour by the agent can be specified in advance. Thus the job of a retail cashier is much more programmable than a software programmer. Task programmability is likely to affect the ease of observing and measuring an agent's behaviour. When an agent's behaviour cannot be easily observed and measured, the problem of hidden action is likely to occur. Arrow (1991) explains:

> "The most typical hidden action is the effort of the agent. Effort is a disutility to the agent, but it has a value to the principal in the sense that it increases the likelihood of a favourable outcome (technically, the distribution of the outcome to a higher effort stochastically dominates that to a lower effort; that is, the probability of achieving an outcome that exceeds any given level is higher with higher effort)." (p. 38)

The agency theory suggests that "the more programmed the task, the more attractive are behavior-based contracts because information about the agent's behavior is more readily determined" (Eisenhardt, 1989a, p. 62). Otherwise, "outcome-based contracts" are more preferable. In the case of software development, the task programmability is low due to the inseparability of design and development, forcing software development to adopt a trial-and-error or an "empirical" process (Section 8.6). This suggests that outcome-based contracts are more appropriate for supplier contracting in ISD projects.

However, outcome-based contracts rely on outcome measurability. When outcomes are difficult to measure (for the lack of time or whatever reason), outcome-based contracts are less attractive. Due to the lack of a full design at the point of supplier contracting, measuring a software product is uniquely difficult. This has been observed in the case studies as the challenges of defining and enforcing acceptance criteria (Section 7.5). The problem of

software product measurement or evaluation is discussed in Section 8.8 as one of the possible root causes for ISD project challenges.

Outcome Uncertainty

Outcome uncertainty has been managed in ISD projects as part of project risk management without much success (Section 7.6). The agency theory approaches the issue by considering how to share the risk between the principal and the agent. According to Eisenhardt (1989a):

> "When outcome uncertainty is low, the costs of shifting risk to the agent are low and outcome-based contracts are attractive. However, as uncertainty increases, it becomes increasingly expensive to shift risk despite the motivational benefits of outcome-based contracts." (p. 61)

Software development displays a high level of outcome uncertainty (Section 8.7). According to the standard agency theory, therefore, outcome based contracts are expensive and less preferable.

Applying the above concepts from the agency theory to software development results in conflicting advice regarding the use of contract types. Due to the lack of task programmability, software development should be contracted based on outcomes. Due to the poor outcome measurability and high outcome uncertainty, the advice is to use behaviour-based contract. This suggests that the agency theory, while useful in providing a conceptual framework for examining possible root causes, is inadequate on its own to provide guidance for an effective solution. Therefore, while analysing each of the possible root causes, concepts and techniques from other theories are also introduced. In particular, the literature on product development (Section 8.3), contract theory (Section 8.5), project uncertainty management (Section 8.7) and product evaluation (Section 8.8) has proved to be helpful.

The focus of this thesis is on the client-supplier relationship. However, other relations within an ISD project may also be construed as PA relationship. In particular, the management team

and the technical team within a supplier organisation may be considered as a pair of PA relation. In this case, the management (the principal) wants to motivate the technical team (the agent) to design and develop a software product. At the same time, the management team wants to control the technical activities so that the delivery time and cost targets are not exceeded. This relation is briefly discussed in the analysis of development uncertainty (Section 8.7). However, the thesis does not go into any in-depth analysis of this or any other PA relations that might be found in ISD projects. They may be explored as future research topics.

8.3 Misconception About Requirements Management

In the idealised Waterfall Model (Royce, 1998), alternatively called "the sequential model" (Kruchten, 2000), requirements should be complete before design and development can start. But that has long been recognised as impossible for software development (Brooks, 1995; Section 7.4.1). The problem of incomplete requirements has been tackled by the iterative software development approach (Stapleton, 1997; Royce, 1998; Boehm, 2000b; Highsmith, 2000; Kruchten, 2000). However, surveys like Standish Group (1995) still point to incomplete requirements as one of the main reasons for ISD project challenges (Section 7.9.2). Conventional advice about outsourcing software development still suggests specifying requirements "in detail" and "thoroughly and precisely" (McConnell, 1997; Wiegers, 2003). This suggests that the iterative approach has somehow failed to overcome the associated challenges. In this section, the current explanation for incomplete requirements adopted by the iterative approach is first examined. Then an alternative is suggested as a conceptual foundation to manage incomplete client requirements.

8.3.1 The Current Explanation for Incomplete Requirements

The proponents of the iterative approach believe that the reason for incomplete requirements for software development is a syndrome known as IKIWISI (I know it when I see it). The

phrase is used to capture the common communication problem where it is much easier to recognise something than to recall from memory (Nielsen, 1993, quoted in Dhamija and Perrig, 2000), not to mention documenting it systematically. The implication of the syndrome is that, according to the proponents of the iterative approach, the assumption that "the requirements are knowable in advance of implementation" is "generally untrue" (Boehm, 2000b). According to Kruchten (2000), users "don't really know what they want, but they know what they do not want when they see it" (p. 56). Kruchten (2000) describes how the iterative approach can overcome the problem of incomplete requirements:

> "[...]address some requirements and some risks, design a little, implement a little, validate it, and then take on more requirements, design some more, build some more, validate, and so on, until you are finished." (p. 60).

The iterative approach assumes that requirements can be tackled "a little" at a time and uses various prototypes to understand requirements (Kruchten, 2000). However, would a client be able to specify what is further required after seeing a prototype? That may depend on both how close the prototype is to the client's desired result, how easily and accurately the client can specify the change(s) etc. Unless what is presented matches exactly what a client wants, there are still difficulties for the client to communicate the changes to be made. In developing a mechanical component, desired dimensions can be specified within a tolerance level. If the component is not correctly produced and requires reworking, accurate measurements can provide exact feedback required for improvements to be made. In other words, not only the initial requirements can be precise, the feedback can be precise as well. The precise feedback derives from the precise initial requirements and the exact measurement of any deviation. Such exact measurements enable accurate communication and understanding between the client and the supplier. The recognised problem of incomplete requirements is equivalent to not being able to specify accurately the desired dimensions for a software product. Due to the lack of a complete and accurate initial requirements specification, feedback from the client to the supplier is likely to be also incomplete. In other words, if the initial requirements cannot be complete for some reason, the feedback is likely to be incomplete for a similar reason,

unless the exact desired product has been obtained and minimal or no improvement feedback is necessary. For this reason, the iterative approach that relies on client feedback may not have been successful in overcoming the root causes of incomplete requirements. The failure of the iterative approach is illustrated by Case 1, where after many iterations using the Rapid Application Development (RAD) methodology, the client disliked the graphic design of the web pages (Section 4.6.3).

8.3.2 An Alternative Explanation: The Kano Model

In search for the root cause of incomplete requirements and possible remedial actions, it is helpful to consider how client requirements are managed in other types of product development. The Kano Model (Figure 8.1, based on Mazur, 1993) illustrates such an approach suggested by the Japanese Total Quality Management (TQM) consultant Noriaki Kano (Mazur, 1993, Cohen, 1995).

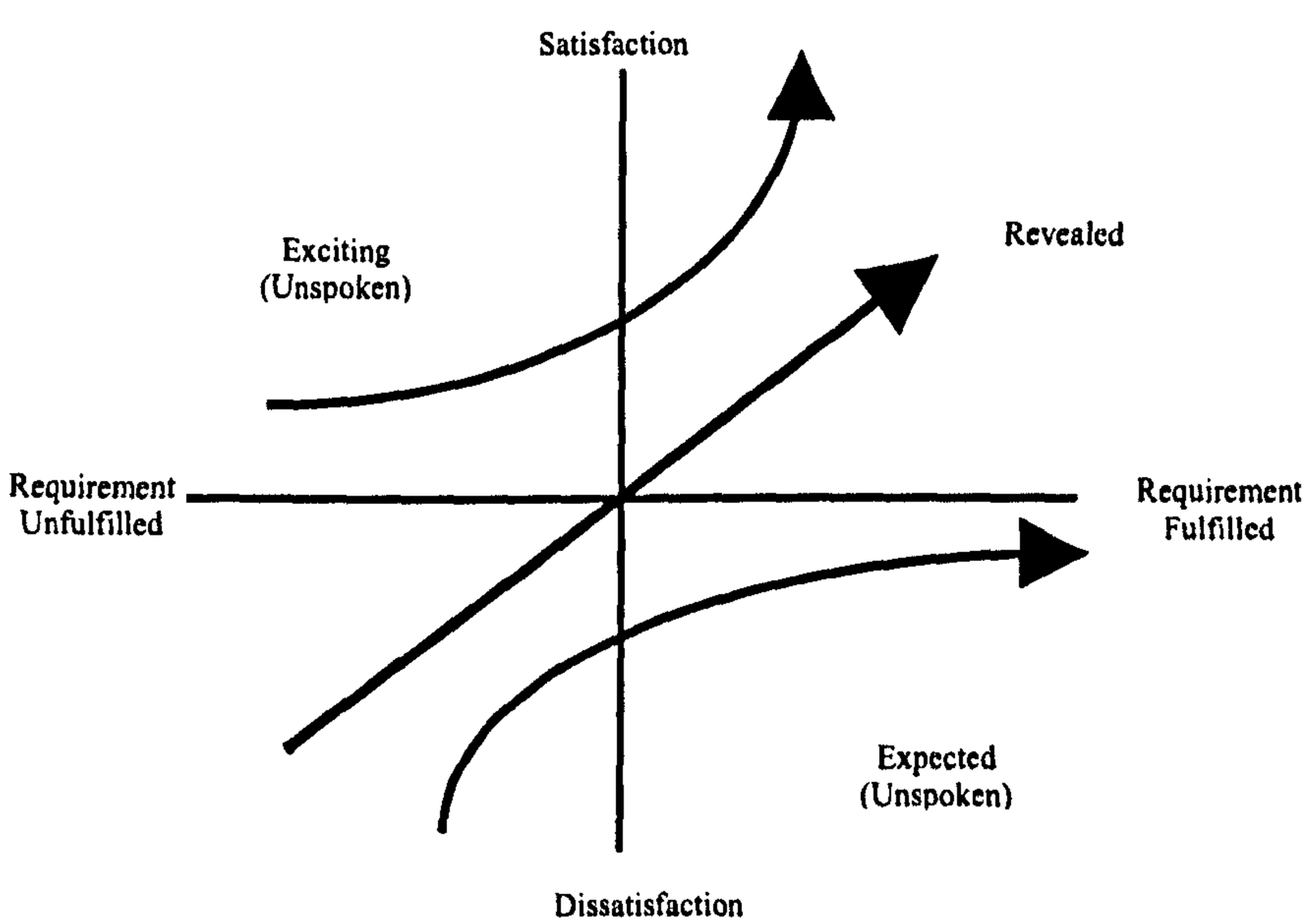


Figure 8.1: The Kano Model

The Kano Model suggests that there are three types of client requirements. Mazur (1993) describes Kano's three types of requirements for the service industries as:

> "**Revealed Requirements** are typically what we get by just asking customers

what they want. These requirements satisfy (or dissatisfy) in proportion to their presence (or absence) in the delivered service. Fast service would be a good example. The faster (or slower) the service, the more they like (or dislike) it.

Expected Requirements are often so basic the customer may fail to mention them – until we fail to deliver them. They are basic expectations of the service, without which the service may cease to be of value; their absence is *very* dissatisfying. Further, meeting these requirements often goes unnoticed by most customers. For example, if an airplane takes off safely, passengers barely notice it. If it fails to take off safely, dissatisfaction, though brief, is intense. Expected requirements must be fulfilled.

> **Exciting Requirements** are difficult to discover. They are beyond the customer's expectations. Their absence does not dissatisfy; their presence excites... Since customers are not apt to be aware of these requirements, it is the responsibility of the service organisation to explore customer problems and opportunities for new levels of service." (p. 2-3)

With regard to a particular business problem, a client organisation may have many needs and wants originated from various sources. These sources can be technical and commercial. They can be objective or purely subjective (e.g. a user' preference for a colour scheme). As illustrated by the Kano Model, some requirements may be explicitly stated by the client, some implicitly expected and some not even known to the client. From this perspective, a complete requirements specification is not possible. The best a supplier can expect is a specification for "revealed requirements". The rest is up to the supplier.

The Kano Model might explain why some of the practices in the iterative software development are problematic. One commonly used approach to manage requirements is for a supplier to document client requirements on behalf of the client or build a prototype and having them "signed off" by the client (e.g. Macromedia, undated). Signing-off would confirm the requirements specification to be full and complete, including the "expected requirements". However, the client is unlikely to be sufficiently informed to do that. As a result, the signature holder goes through a lengthy consultation process as a rear-guarding exercise. This causes delays in signing off requirement specification, which is a common complaint from software suppliers (Section 4.6.1). The delays caused by sign-offs, however, consume precious project times, which may add to project delays.

The Kano Model also helps to resolve the debate regarding whether a software product should satisfy the client's requirements or the client's expectation. The answer is "both". The client only manages to specify the revealed requirements but expects the delivered product to have the "expected requirements" built-in. The misconception that a client should "document the requirements thoroughly and precisely" (Wiegers, 2003) is asking the client to do the impossible. The iterative approach recognises the fallacy to a certain degree. Unfortunately, it is easy to fall into the trap of the misconception (e.g. Section 4.7.5). The Kano Model can guide ISD projects to adopt a more realistic approach to client requirements management. This is incorporated in designing the potential solution in Chapter 10.

8.4 Switching Cost and Its Impact on Supplier Behaviour

Switching cost is the cost a client incurs or perceives to incur when changing from one supplier's product to that of a competing supplier's, even when both products are functionally the same (Klemperer, 1995). It can be informational, transactional, artificially created, learning-related or psychological (Klemperer, 1987, 1995). When switching cost is so high that switching suppliers is unthinkable, the client is said to be locked-in with the existing supplier (Varian, 2001). Switching cost and supplier lock-in is widely recognised in IT industry and the software market (Varian, 2001). NAO (2001) provides the NIRS Project as an excellent example of switching cost that prevented the client from changing the supplier. In this case, the client had to decide how to purchase additional system functions. The choices were either to stay with the incumbent supplier or to contract with a new one. The client found that switching the supplier would have incurred a £44million contract cancellation fee for a contract originally worth up to £76million. Not surprisingly, the client decided to stay with the incumbent supplier. The £44million is clearly a part of the total client switching cost. The case serves to illustrate the potential high level of switching cost in the software market.

Switching cost may affect competitive behaviours of market participants leading to price wars in the pre-contract bidding stage (Klemperer, 1989). Consider a two-period market with a switching cost *s* ($s > 0$) and the real cost of the product *c* including the supplier's normal rate of return, the supplier's bidding price P in the first period could be as low as (c - s). In the second period, the supplier charges higher prices to recover up to the value of s. In the case of lock-in, the supplier charges monopoly prices. Since *s* is high in acquiring software products, *P* is likely to be far less than the supplier's real cost, resulting a significant bargains-then-ripoffs opportunistic behaviour pattern (Klemperer, 1989, Farrell and Klemperer, 2001). One dramatic example of this opportunistic behaviour is found in the Case 2. The supplier AES requested 40% price increase once selected and agreed to 20% increase after tough negotiations (Section 5.5.2). In Libra Project (NAO, 2003), the supplier ICL increased the bid from the initial £146million to £184million before the contract was signed, an increase of 26%. The concept of switching cost provides a good explanation for these price increases.

One problem for suppliers in the stage of contract bidding is that a client's switching cost may or may not be observable. The switching cost is said to be observable if it is public information. When that is the case, all suppliers will be bidding with the same level of *s*. In the software market, however, a client's switching cost is hidden and not easily observable. Suppliers would have to rely on their own estimates as part of bidding price considerations. Thus it is possible for suppliers to over estimate *s*, increasing the possibility of giving unrealistic price commitments (e.g. Cabinet Office, 2000).

The initial bargains present the client with a supplier selection problem. The prices offered by suppliers no longer simply signal their competitiveness. They also incorporate the suppliers' hidden estimates of the clients' switching cost. The more opportunistic a supplier is, the lower the quoted price might be. To make the matter worse, a client is often obliged to select the supplier with the lowest bid, especially in the public sector where strict guidelines are in place

regarding the purchasing process. This leads to adverse selection (Section 8.2) and was believed to contribute to ISD project challenges and sometimes failures (e.g. the London Ambulance Service's Computer Aided Despatch system project in Page et al., 1993).

The above discussion assumes that quality is held constant across the two transaction periods. However, like price, quality commitment may be affected by the switching cost due to incomplete information (Section 8.2). There is a tendency that a supplier claims that its product quality is superior than it may be. One technique used by software suppliers is the "Bait and Switch Game" (Johnson, 2000a). A supplier would send the "A" team of highly skilled and experienced professionals with great credentials, but delivers with the "B" or even the "Z" team. Such a game is aimed at giving a quality image first and then attempting to get away with lower quality counting on the client's switching cost. This game was played by NQ in Case 1. NQ's proposal document was of high quality, led by one of the firm's partners. Cost information shows that the partner's involvement stopped by the end of June 2000 when the contract was signed. NQ also promised system testing "against defined test scripts written following design" but refused to carry out such testing without extra funding later (Section 4.6).

Suppliers' bargains-then-ripoffs behaviour pattern mirrors the transition of the market structure from being competitive before the supplier selection to being monopolistic thereafter. According to the neo-classical economic analysis of Structure, Conduct and Performance (SCP), structure determines performance (Ferguson and Ferguson, 1994), not having to consider conduct (behaviour) at all. If we accept SCP theory, we may come to the conclusion that bidding output (the bidding document, the pledged price, the pledged product and the pledged time of delivery) will be excellent due to the competitive market structure. On the other hand, the output from the development phase (the scope and quality of the actual product, the actual delivery time etc) will be poor and the actual price will be higher (the

monopoly price) than the promised one (the competitive price). The observations on ISD projects bear this analysis out. The reduced functionality, poor quality of the delivered product, budget overruns and late delivery are indeed the symptoms of ISD projects.

In summary, though suppliers are often selected and contracted on a competitive basis, the product design and development is typically carried out by a single supplier as a monopoly. The high switching cost associated with acquiring software products leads to distorted bid prices, which is made worse by hidden information between the client and the supplier. The overall result is a bargains-then-ripoffs opportunistic behaviour pattern on the supplier's side. This pattern not only may affect product price but also product quality. The SCP analysis provides further theoretical support to the explanation provided by switching cost. It also suggests a direction for resolving some of the ISD project challenges by introducing competition into product development, thereby reducing client switching cost for possible alternatives. This will be incorporated in the unified explanation proposed in Section 8.11. In the meantime, it has to be admitted that there are successes as well as failures in ISD projects (Standish Group, 1995) operating with the same market process. Such different results have to be explained, possibly by the conduct of firms, influenced by other factors. These are further analysed in the subsequent sections.

8.5 Incomplete Contracts

In an ISD project, a client acquires an application software product by offering a contract with a selected supplier. Inadequate contracts have been cited as one of the reasons for ISD project challenges and failures (Webster, 2000). Webster's recommendations are clear:

> "All parties [...] should agree ahead of time to specific expectations, promises, and contingencies regarding each of the areas of quality [...]. For example, the system specifications should include not just the required functionality, but should also spell out any performance requirements or constraints, compatibility requirements, anticipated lifespan, and acceptable levels of defects."

Webster raises the problem of "incomplete contracts" in ISD projects. The problem of

"incomplete contracts" might have explained many of the challenges of supplier contracting and subsequent software developments. Indeed, without exactly agreeing "specific expectations, promises and contingencies", how could a supplier know what to develop and how could a client's needs be met? But it does not appear to be as simple as that. The following subsection examines the concept of "incomplete contracts", followed by Section 8.5.2 on contracts used in ISD projects.

8.5.1 The Concept of Incomplete Contracts

A contract is incomplete if the goods to be exchanged are not described in full *ex ante* for a third party (e.g. the court) to enforce (Hart and Moore, 1999). Maskin and Tirole (1999) refer to such a problem as contracting parties having "trouble foreseeing the possible *physical* contingencies" (emphasis in the original). This appears to be common in design-build contracts due to a phenomenon called "preferential engineering", where a client influences the design by imposing its own preferences. As a result, "liabilities may become confused" (Marshall, 1992). Contracts with application software suppliers are typical examples of incomplete contracts since software products are difficult to describe in full before they are developed. This can be attributed partly to the problem of incomplete requirements in the form of functional specifications, bearing in mind the Kano Model (Section 8.3). In addition, design and build are difficult, if not impossible, to separate in software development contracts (Section 8.6). The nature of software contract incompleteness is demonstrated by the contract between the Lord Chancellor's Department and ICL in the Libra Project reported in NAO (2003). Though the supplier in this case failed to supply the promised software, the client did not feel that it could take the supplier to court for fear of "claims and counter claims" (Computing Staff, 2003). This might be explained by the concept of incomplete contracts in that what was agreed between the contracting parties (presumably for the supplier to sell a software product to the client) was felt not enforceable by the court.

Within an incomplete contract, price and quality commitments from a supplier are related. Farrell and Shapiro (1989) study how a seller may behave if an incomplete contract constrains some dimensions (e.g. price) and not the others (e.g. quality). They arrive at the following conclusions:

1. If the seller is expected to exploit the buyer in any case *ex post*, then it is better for the seller to do so unconstrained. Otherwise if the buyer constrains the seller in some variables (e.g. price) and not others (e.g. quality), the result may be inefficient in that the buyer will get the goods meeting the constrained variables (e.g. price) and is exploited on the unconstrained variables (e.g. quality). This is the "Principle of Negative Protection";

2. It can be desirable to make the seller's exploitation inefficient, if it means that he will then refrain altogether. That is to say, if a contract can be designed and accepted by the supplier so that its exploitation (lower quality) will cause inefficiency to itself, then such a contract will produce efficient result. This is the "Principle of Positive Protection".

In the case of contracting for application software in ISD projects, some dimensions (e.g. price and delivery dates) are easily contractible and others (e.g. functions and quality) are not. According to Farrell and Shapiro's first conclusion above, therefore, it may be better to have no contract at all than to have a partial contract. In Case 2, the client and the project team did not like an initial proposed delivery date of 02/07/2001 for "Release 2". The supplier duly promised to bring the delivery date earlier. The "acceptable" date and a fixed price were written into the contract (Section 5.5). What actually happened was that "Release 2" was redefined by the supplier in such a way to have only contained a revised "Release 1". The original "Release 2" was not delivered until 14/09/2001, some two and half months later and did not function as promised. This serves as an example for how a supplier might be willing to commit on easily contractible dimensions according to the client's wishes and exploit the unconstrained dimensions.

The second conclusion above calls for contracts to be designed in such a way that they will "hurt" the supplier should it pursue actions of exploitation. Maskin and Tirole (1999) suggest that, when it is difficult to specify "physical contingencies", a contract might be written to

specify "the possible *payoff* contingencies" to overcome the problem of incomplete contracts. This effectively calls for outcome-based rather than behaviour-based contracts according to the agency theory (Eisenhardt, 1989a; Walsh and Schneider, 2002; Lichtenstein, 2002). Though in Section 8.2.2, it has already been demonstrated that neither behaviour-based, nor outcome-based contracts are effective on their own, the outcome-based contract type with "payoff contingencies" might offer a possible solution on the whole. This is adopted in designing the proposed management framework in Chapter 10.

Another perspective to examine client-supplier contracts is to see if they meet the following two constraints (Anderhub et al., 2000):

- *Incentive compatibility constraints*: this is necessary so that the interests of the principal and the agent are aligned;
- *Participant constraints*: this is necessary so that the agent will agree to the contract.

Anderhub et al. suggest three steps in a simplified PA contracting process. First of all, the principal offers a contract with a fixed wage and a return share. The fixed wage can be positive or negative and the return share can vary between 0 and 100 percent. An example of the negative wage is the rent a tenant pays to the landlord. A return share of 0 percent from the principal to the agent means that the principal keeps all the return while 100 percent means that the agent possessing the whole return. Secondly, the agent decides whether to accept the contract or not. Thirdly, the agent chooses an effort level that generates a return and causes a cost (to be borne by the agent). The landlord may offer a contract that demands a fixed rent to be paid and leaves the farmer 100% of the residual return. Should the contract be accepted, the farmer will exert utmost effort to produce as much residual return as possible. The two criteria and the three steps provide a basic conceptual framework to analyse contracts below. In addition, an attempt to adopt an outcome-based contract in Case 3 is examined to see why it might not have been successful.

8.5.2 Contracts Used in ISD Projects

There are two types of contracts commonly used in ISD projects: time and materials (T&M) and fixed price, both observed in the case studies. With T&M, the client bears all the risks since the supplier is paid according to the amount of time worked. Therefore, T&M contracts meet the criterion of participation constraint. As for the incentive compatibility constraint, it fails altogether. While the client's objective is to complete the project as budgeted or sooner, the supplier might be rewarded for taking longer to complete the required tasks. When the budget ceiling is hit, the supplier simply stops making effort and demands an increase to the budget. The alternative to that is for the client to accept an incomplete product or a lower quality one. Thus a simple T&M contract introduces a disincentive to complete the development early or deliver a quality product. The T&M contract with NQ in Case 1 might be seen as one of the direct causes for the poor quality web page design (Section 4.6.3).

While T&M contract leaves both the price and the product quality unconstrained, a fixed-price software contract constrains the price without having effective means to constrain product quality. Fixed price offers the supplier a fixed wage and 0% return. For software products, though difficult, agreeing fixed prices is possible since cost can be monitored relatively easily. Unfortunately, contracting on quality is much more problematic (Section 7.5; Section 8.8). In fixed-price contracts, therefore, the supplier tends to exploit on the quality dimensions other than the price (Farrell and Shapiro, 1989). As a result, a fixed-price contract does not produce desired software products. The contract in Case 2 illustrates the point. In Case 2, the price was fixed after the initial shock increase (the effect of the switching cost – Section 8.4). The acceptance criteria were only available in general simplistic terms (Section 5.5). The contract might be improved upon in various ways but for the client to insist on detailed quality description without having seen the product would have been difficult if not impossible.

While it might be impossible for a client and a supplier to describe and agree "physical contingencies", would it be practical to adopt Maskin and Tirole's (1999) suggestion to describe the "payoff contingencies" so that an outcome-based contract could be adopted? The contract with SC1 in Case 3 made a limited attempt to do just that. Though no detailed description of contingencies were available linking with payoffs, there was a statement about the "maximum loss recoverable" by the client being 100% of the contract price (Section 6.5). This describes at least the possibility of the software product being rejected. In other words, it might be interpreted that the contract envisaged a number of payoff contingencies on a spectrum. On one end of the spectrum, the client would accept the software product and the supplier would be paid in full. On the other end, the client would reject the product and the supplier would be not paid at all. The actual result of the project was neither an outright success, nor an outright failure. Even so, theoretically the situation must have been covered by the payoff contingency spectrum. Yet no record exists to indicate that the client recovered any loss from the supplier. In fact, it is not easy to enforce the clause between the client and the supplier. Apart from the possible difficulties with measuring the outcome of the software product (Section 7.5; Section 8.8), the client in this case faced a problem of having no practical option to meet the regulatory requirements. Should the client decide to recover the whole amount from the supplier, it would have to abandon the software product without meeting the regulatory requirements. Going for another supplier's product at that stage would have meant incurring the switching cost. The fact that the client planned to extend the project for another year investing an extra £996K (Section 6.5) indicates that the client's switching cost was too high to have taken any other alternative action.

In summary, neither T&M nor fixed price contract types can effectively constrain software product development due to the difficulties of completely specifying requirements and controlling quality dimensions. The use of payoff contingencies in contracts must be backed up with realistic options, which are limited due to high switching costs. This points to a

fundamental problem with monopolistic software development inherent in the current approach of managing ISD projects.

8.6 Inseparability of Design and Development

It is a common assumption that software design can be completed fully before development (Kruchten, 2000). This assumption is misplaced based on the following observation:

> "You can accumulate pages and pages of design documentation and hundreds of blueprints and spend weeks in review, only to discover, that in the process, that the design has major flaws that cause serious breakdowns" (Kruchten, 2000, p. 57)

This observation reveals a fundamental feature of the software product development process, which might be called "the inseparability of software design and development". This is related to the nature of a software product as "pure design" (Callahan and Moretton, 2001; Eischen, 2002; Yeargin, 2002). While a construction product may be designed first and its quality controlled at the virtual level before production (building) work starts, software design is intertwined with development. A software design on paper cannot (at least at the current level of software technology maturity) be certified to be "right" with the necessary quality dimensions of "correctness, efficiency, feasibility and so on" (Kruchten, 2000, p. 57) without being tried out first by a stage of development. The process of development can reveal areas of incorrectness and inefficiency of the design. Therefore, the software development process might be described as a "trial-and-error" process. Williams and Cockburn (2003) refer it as "an empirical process" characterised by "inspect-and-adapt" cycles.

The trial-and-error approach may explain observations of software products taking longer to develop, causing cost and schedule overruns. This is because the number of cycles of the iterations between design and development are not easy to predict beforehand. The trial and error process might render a previous endeavour wasted. Royce (1970) warns that a software development may return "to the origin", causing "100-percent overrun in schedule and/or cost" (p. 2). The inseparability of design and development may also explain the difficulties in

co-ordinating designers and developers. A designer is not able to finish a design in isolation of the developer or vice versa. McBreen (2002a) calls for integrated teams and recommends that "designers should be deeply involved in the programming tasks".

The inseparability of design and development may also explain the poor documentation in the process of software development (Section 7.5.4). Nayar (2000) suggests that poor software documentation is due to the misconception that "it is a support task that is essentially noncreative and a necessary evil [...]" (p. 29). There might be some truth in that. An alternative explanation, however, is that design and development are closely related so that design documents become redundant once coding is started. A design may be documented first and then "implemented" through development. The development is likely to reveal the inadequacy of the design. At that point, the development team has two options. The first one is to modify the design document and then to implement it to see if it works. The alternative is to modify the developed software directly. Since the modified design is not guaranteed to work and may require further changes, it stands to reason that the second option is likely to require less time to obtain the necessary feedback for further adjustments. In that way, while initially a design may be documented, it can easily become obsolete as soon as software development is started. As the software product evolves, the design evolves without necessarily being updated separately. In other words, design and development activities are too closely related with each other to encourage separate documentation.

While the trial-and-error approach has *always* been the reality for software development (see e.g. GenerExe, 2002; Lindberg, 2003), it has often been regarded as a problem (e.g. Raybould, 2002). However, the proponents of the iterative approach have recognised the inseparability of design and development and have suggested various incremental and iterative techniques to address the problem (McManus, 1997; Royce, 1998; Kruchten, 2000; Larman and Basili, 2003). For example, Beck (2000) advocates having "enough design for

today's code" (p. 65). In other words, do not aim for a complete design in the first place. Instead, design enough for the immediate development needs and design more after coding to see how best to proceed. However, the iterative approach does not deal with a number of other project challenges and their other root causes adequately. For a fuller critique of the iterative approach, see Section 8.11.

8.7 Risk and Uncertainty

The observations from the case studies suggest that the risk management process may be effective dealing with certain risks but not others (Section 7.6). To understand the possible reasons behind such observations, it is helpful to examine the concepts of risk and the related concept of uncertainty (Section 8.7.1). It is suggested that uncertainty with software development is an important root cause for estimating errors and other management difficulties (section 8.7.2).

8.7.1 The Concepts of Risk and Uncertainty

In the literature about project management and risk management, there are a number of different approaches to defining risk and uncertainty and the relationship between the two (Williams, 1995). For example, Hertz and Thomas (1983) designate risk as "both uncertainty and the results of uncertainty" (p. 3). The distinctions between risk and uncertainty, according to Hertz and Thomas, "have limited value in the practical process of risk assessment and analysis. Likewise, Kerzner (1998, p. 869) defines risk as a function of uncertainty:

Risk = f(event, uncertainty, damage)

The implication of this approach is that risks can be "identified, assessed and provided for" (Kerzner, 1998, p. 871). The risk management process is therefore a process of reducing uncertainty associated with risks by identification, quantification, evaluation, prevention, reduction and transference etc (e.g. CCTA, 1998; Kerzner, 1998, Wiegers, 1998; Lieberman, 2002).

Alternatively, Chapman (2000) uses the term "project uncertainty management" to denote "an even-handed approach to opportunities and threats" in contrast to "project risk management" with the focus only on "threat management" (p. 241). This implies that "uncertainty" is a broader term than "risk" and risk is merely a subset of uncertainty. Yet another approach treats risk separately from uncertainty. This is represented by Correia et al. (1989, quoted in Remenyi, 1999, p. 7):

> "uncertainty implies that either all the alternative possible outcomes cannot be identified, or that no probability can be attached to the alternative possible outcomes."

In this sense, uncertainty has been referred to as "unknown unknowns" while risk is "known unknowns (e.g. Crawford, 2003). The implication is that uncertainty cannot be managed with the usual risk management techniques of identification, quantification, evaluation, prevention, reduction and transference etc. Where such a distinction is made, the focus tends to be on the management of risk rather than uncertainty (e.g. Remenyi, 1999; Thomas, 2000).

In this thesis, the definitions of risk and uncertainty are distinguished as suggested by Correia et al. and the focus of the following discussions is on the management of uncertainty. Turner and Cochrane (1993) suggest four types of projects based on two parameters: how well defined are the goals and how well defined are the methods of achieving them. They suggest that, for software development, goals are not well defined while methods are (Type 3 Project in Turner and Cochrance, 1993). In fact, research evidence has suggested that in reality neither goals nor methods are necessarily well defined. Ziv et al. (1996) identify many sources of uncertainty in software development and proposes an "uncertainty principle", stating that "uncertainty is inherent and inevitable in software development processes[...]". This suggests that the methods (i.e. the processes) are poorly defined. Schwaber (1996) describes the systems development process as "unpredictable". McBreen (2000) asserts that "[u]ncertainty is one of the few certainties about software development". Uncertainty poses

serious problems to software development, rendering it more like research (O'Neil, 1999), which is Type 4 Project in Turner and Cochrane (1993). In the following subsection, the development uncertainty is highlighted to see how it might affect a software supplier's commitment to deliver a quality software product within cost and schedule. Other types of uncertainties as suggested by Ziv et al. (1996) might be explored for future studies.

8.7.2 Software Development Uncertainty and Its Consequences

Software development uncertainty might be described in simple terms as follows. A seemingly simple technical task may lead into a number of other tasks totally unforeseen. When a non-trivial software program is written, the programmer cannot be sure if it will be compiled without error. Nor can the programmer be sure that it will execute correctly as intended despite the programmer's *best endeavour*. Here the issues of staff motivation and process control are not relevant. In other words, there is uncertainty involvement in software development, which is technical in nature, not managerial.

The uncertainty contributes to the need for the trial-and-error approach discussed in Section 8.6, resulting necessary development "restarts" (Royce, 1970) and making the software development process "unpredictable" (Schwaber, 1996). This may explain the estimating difficulties associated with software development. Estimating effectively attempts to predict a level of required man-days (or man-months) for accomplishing a particular task or a developing software product. Yet if design and development involves uncertain number of iterations and indeed uncertain number of "restarts", the estimate is likely to be inaccurate. In studying software estimating accuracy, Lederer and Prasad (2000) conclude that neither the "Informal Basis" nor the "Algorithmic Basis" improves estimating accuracy. More interestingly, "revising an estimate does not appear to make it more accurate" (Lederer and Prasad, 2000, p. 41). Development uncertainty and the inherent trial-and-error approach might explain the failure to provide accurate estimates for software development.

Uncertainty may also explain why software development is successful sometimes within the ISD project management framework but not others (Standish Group, 1995). In a particular project, there is a software development team (typically in the software supplier's organisation within the context of this thesis) working on design and development. At the same time, the supplier management tries to manage the development and delivery of a software product within a budget and a deadline according to its initial estimate and promise. The client's project team is charged with the responsibility to acquire the software product with a specific budget and deadline of its own. At a technical level, the software development team may work hard to iterate design and development activities in resolving technical issues, and when necessary restarts with new designs for better results. Thus on the one hand, it is not possible to predict when a software product can be successfully developed due to the trial-and-error process. On the other hand, both the supplier management and the client project management team try to manage cost and schedule tightly.

There may be many possible outcomes as a result of the conflict between development uncertainty and tight project management. Four possibilities are considered here. The first is that the design and development takes place as initially estimated. As a result, cost and schedule commitments might be kept and the resultant software product is of the expected quality. This scenario has not been observed in the three case studies but it may have been achieved by the Computerisation of PAYE Project (Morris and Hough, 1987). The scenario may also describe projects in the "successful project" category reported by Standish Group (1995). The second possibility is that an initial design is broadly "right" but requires considerable number of development iterations to be improved. This may results in cost and schedule overruns but the software product may be of acceptable quality. The OMSTool in Case 3 might fit into this scenario although it appeared to need an additional year to achieve what had been planned initially. The third possibility is that after a number of iterations the

budget or deadline may be reached and development stopped with an incomplete software product delivered. This was observed in Case 1 with NQ's deliverables (Section 4.6). The fourth possibility is that after unexpected number of design-development iterations and possible restarts the software product is still not "right". But due to project management pressure, the resultant software product is delivered and implemented. This scenario results in project cost and schedule overruns and poor software product functionality and quality. This was observed in Case 1 (Section 4.4) and Case 2 (Section 5.8).

8.7.3 The Management of Uncertainty in Projects

As suggested in Section 8.7.1, some project risk management literature chooses not to discuss the management of uncertainty even when it is recognised to be different from conventional risks (e.g. Remenyi, 1998). When uncertainty is discussed, the suggestion is usually to make available a level of contingency funding (e.g. Schulte, 2002; NIH, 2000). However, there are at least two problems with the contingency funding approach. The first is the determination of the funding level, which cannot be guaranteed to be sufficient *ex ante*. The second problem is that the extra funding may provide a partial solution to the problem of cost but it cannot resolve the fundamental problem of product quality. Consider Case 2 as an example. The project team did prepare a contingency budget. It was less than 10% of the original project budget (Section 5.9.6). Though it would have helped to extend the project by a few weeks it could not have changed fundamentally the outcome of the project.

Project uncertainty does not reconcile easily with the usually fixed project cost and schedule targets. A possible solution is to increase project flexibility and to remove the cost and schedule restrictions altogether. In software development, this flexibility approach is adopted by the "SCRUM methodology" for software development as proposed by Schwaber (1996):

> "The estimate is only for starting purposes, however, since the overall timetable and cost are determined dynamically in response to the environmental factors." (p. 18)

The SCRUM methodology might be applicable to suppliers developing software products for the PPSW market. For ISD projects, software suppliers usually work with strict budget and time commitments. This was the case for the fixed price supplier contracts in both Case 2 and Case 3. There are great resistance to "dynamically" adjusting the cost and time commitments in project environment. Clients naturally feel suspicious about any such adjustment request from the supplier in a monopoly setting. In Case 1, the "re-planning" exercise allowed MSS to adjust its cost. The SLC project manager suspected that MSS had increased the estimates by 20% more than necessary (Section 4.7.6). In the Libra Project reported in NAO (2003), the supplier ICL's commitments on cost, schedule and deliverables were "dynamically" changed a number of times, each change increased the average annual cost significantly (Table 8.1, based on NAO, 2003). The development part of the contract, which was to be about two thirds of the contract value, was eventually cancelled (NAO, 2003). These observations indicate that flexibility with cost and time commitments within the current monopolised development framework potentially results in significant cost increases without necessary helping the development of a quality product.

Table 8.1: Dynamic client-supplier contract changes in the Libra Project

Date of Agreement	May 1998	December 1998	May 2000	July 2002
Contract Cost (£million)	146	184	319	318
Contract Length (Years)	11	10.5	14.5	8.5
Average Annual Cost (£million)	13.3	17.5	22	37.4

In the search for an effective strategy to manage project uncertainty, Pich et al. (2002) provide a "complete" model summarising the following three approaches:

- The instructionist approach: pre-specifying and triggering actions based on signals;
- The learning approach: conducting new planning in the middle of a project;
- The selectionist: the pursuit of multiple candidate solutions until the best can be identified.

Pich et al. suggest that the common project management techniques of task scheduling and risk management with contingent action are the hallmarks of the instructionist approach. This approach does not work when there is inadequate information due to "project ambiguity" (events or causality being unknown) and "project complexity" (an inability to evaluate the effects of actions because too many variables interact). The learning approach allows active incorporation of new information, either "incrementally" or "opportunistically", thus overcoming to some extent the problem of inadequate information. However, learning is merely an extension of the instructionist approach. It relies on the identification of the optimal policy to be modified overtime. The alternative is to try multiple solutions in the hope that one will work. This is the selectionist approach. This approach has been studied and practised to address development uncertainty in different forms. Abernathy and Rosenbloom (1969) recommend "a parallel strategy" for research and development projects. Sobek et al. (1999) formalises a "Set Based Concurrent Engineering (SBCE)" framework based on Toyota's successful practices for manufacturing product development. SBCE provides a good example of the selectionist approach at work (see Section 9.5 for a comparative study between software development and manufacturing product development).

In software development, the Waterfall Model might be said to follow the "instructionist approach". This has long been recognised as not suitable for software development (Royce, 1970; Larman and Basili, 2003). Many current ISDMs are based on the iterative approach, which might be equated to the learning approach (Schwaber, 1996; Stapleton, 1997; Royce, 1998; Kruchten, 2000; Beck, 2000). Based on the observations from the case studies, the learning approach does not resolve the uncertainty problem effectively either (for a fuller critique of the iterative approach see Section 8.10). It is suggested that the selectionist approach be adopted for software development (Section 8.11 and Chapter 10).

8.8 Product Evaluation

In ISD projects, there are two stages when product evaluation is conducted. At the pre-contract stage, a client selects a supplier by evaluating partially available products (prototypes or COTS if available) together with suppliers' promises. At the product acceptance stage, the client evaluates a delivered product as a quality control measure. It is suggested in this Section that there are problems associated with both stages of evaluation. Before examining these problems, two possible evaluation approaches are discussed first below.

8.8.1 Two Approaches of Product Evaluation

A product might be evaluated by comparing it against a complete model or with other comparable products. The former is called the absolute product evaluation approach (Morisio et al., 2002; Blin and Tsoukias, 2001) and the latter the relative one. The absolute approach is common in verifying manufactured goods. Consider a mechanical part going through an inspection process as an example. The complete model is embodied in one or a series of drawings and specifications that are controlled by the client. A client's quality inspector can tell if any dimension does not match what is stated in the specification by using appropriate measuring tools. The measurement can be repeated within an expected level of tolerance and diagnosis for any incorrect dimension can usually be agreed between a client and a supplier without much dispute if the measuring tools are in good order. Such a process cannot be easily applied to the evaluation of a software product. First of all, the client, when commissioning software development, does not possess a complete design. In fact, such a design is one of the key deliverables the client is looking for from the supplier due to the inseparability of design and development (Section 8.6). The client only has a set of incomplete requirements, which is inadequate in controlling the eventual software quality. Secondly, there are no readily available and objective measurements for many software dimensions. Consider Case 1 for example, in terms of functional requirements, how many strokes should a user key in to complete an online purchase transaction? In terms of non-

functional requirements, how should usability and level of security be measured? These are open questions without clear-cut answers. Thirdly, where measurements are easily obtainable (e.g. performance measured in time), it is not always repeatable. Even when a measurement is obtainable and repeatable, it is often difficult to diagnose the root causes for undesirable results. An unfavourable evaluation result can be interpreted in many ways. Is it an unknown application software defect or is it a result of changed system software configuration that causes an application to crash? Is it the client's defective data or the network traffic, or some mishandled memory allocation within the system software that causes the application performance degradation? In Case 3, the client and the supplier spent months wondering if the poor system performance was due to some network problems or due to application defects without being able to reach an agreement (Section 6.9). Clearly, the absolute evaluation approach can be slow, difficult and costly.

The relative approach is generally known as benchmarking and it can be used to measure any conceivable dimension of products, services and work processes (Spendolini, 1992). It is common even for organisations to benchmark themselves against others to gauge their own performance. The enthusiasm for such cross-organisational benchmarking can be partly explained by the fact that the absolute evaluation on one single instance is difficult and costly. Benchmarking has been known in many other ways as relative performance measurement (RPM), relative product evaluation (RPE) or any number of methods with the same approach. RPE has been widely used by trade magazines to compare one product against another (e.g. Loverica, 1994; Henning, 2001). In ISD projects, RPE has been used often to evaluate COTS products (see e.g. Kontio, 1996) prior to supplier selection, as happened in the feasibility stage in Case 2 (Section 5.3) and the detailed supplier selection exercise in Case 3 (Section 6.4). What can be established is that the relative approach is widely applicable and intuitive. However, the prerequisite for using the relative approach is to have two or more comparable products, services or processes to compare with. Thus the method has not been used in ISD

projects at software acceptance stage. The current single source software development framework forces the client to adopt the absolute evaluation approach, which might explain the problems with enforcing acceptance criteria (Section 7.5).

The absolute and relative approaches are not mutually exclusive. A combined approach may be taken as a matter of common sense. For example, in evaluating Intel's Pentium 4 2GHz CPU, Henning (2001) lists many performance measures in absolute terms and then compares the Intel's CPU with three of AMD's CPUs. At the software acceptance stage, a combined evaluation approach is likely to assist the client to make an informed decision.

8.8.2 Two Stages of Product Evaluation in ISD Projects

The two stages of product evaluation in ISD projects are contrasted in Table 8.2. The table illustrates different problems for each of the stages. While the pre-contract evaluation takes the relative approach, it is based on suppliers' partial products (prototypes and reference installations if available) together with promises of price, quality and delivery time etc. Case studies have shown that partial products or prototypes are poor indicators to the eventual outcomes of software products (e.g. Adolph, 2000). Even reference installations cannot guarantee that the same software application for one organisation will be successful for the other (Mitev, 1996; Southon et al., 1997). Past performance, company reputation, personal experience with suppliers may all feature to a certain degree in the client's eventual decision but none gives a sufficient indication of a final software product's success, which might be explained by development uncertainty discussed in Section 8.7. Many case studies have identified that the supplier selection process is problematic (e.g. Page et al., 1993; West Essex RISP Project in Flowers, 1996; Collins, 1998). Even with an impeccable process, the inherent weak basis for product evaluation together with development uncertainty might still leave the client with a failed end product.

Table 8.2: Two stages of product evaluation in ISD projects

	Objectives	Basis of Evaluation	Approach	Problems
Pre-Contract Product Evaluation	Supplier/Product Selection	Partial products and/or promises	Relative	Weak basis for predicting product success
Post-Contract Product Evaluation	Product Quality Control: for acceptance	A single "finished" product	Absolute	No absolute model to evaluate against. Difficulties to enforce acceptance criteria

In contrast, in the current ISD project framework, the post-contract product evaluation is based on a single software product. This forces the client to take the absolute evaluation approach. Theoretically, the client has three options. One is to accept the software product with no further action required from the supplier. Either the product meets or exceeds the requirements, or the client is willing to leave any functional or quality gap to be filled later. The second option is to allow the development to continue if the client is not satisfied with the product, thus extending the overall project. The third option is to cancel the software development, and to source a new software product or to cancel the project altogether. Section 8.5 has already discussed the lack of realistic options in practice. This is made more difficult by not having a complete model to evaluate the software product. The product acceptance evaluation therefore fails to achieve its objective of controlling product quality.

8.9 The Traditional Systems Development Framework

Sections 8.3 to 8.8 have discussed various problems in managing application software suppliers. These problems include the misconceptions about incomplete requirements, the high switching cost in software acquisition, the incomplete contracts, the inseparability of design and development, the development uncertainty and the absolute production evaluation approach. These causes interact with each other closely to produce the pattern of ISD projects challenges. For example, the problem of incomplete requirements leads to incomplete contracts due to the misconception about the contracting strategy. The high switching cost, inseparability of design and development together make it possible for the supplier to pursue

a bargains-then-ripoffs opportunistic behaviour pattern. Development uncertainty and its associated trial-and-error process leads to estimating inaccuracies, which makes it genuinely difficult for the supplier to control development progress. Due to the cost implications, the supplier is likely to limit iterations. Instead, it is likely to exploit on product quality dimensions, which is possible due to incomplete contracts and the limitations of the absolute product evaluation approach. Despite many "best practices" and ISDMs, including those based on the iterative approach, ISD projects seem to proceed in some common framework and often end up with cost and schedule overruns and poor software systems. In this section, this common framework is re-visited in light of the concepts and issues discussed in Section 8.2 to 8.8. Section 8.9.1 attempts to describe this common framework, referred to as the traditional systems development framework (TSDF). The step-by-step description paves the way for a game-theoretic modelling for TSDF in Section 8.9.2 (cf. Section 10.9).

8.9.1 Describing TSDF

This Section describes TSDF by tracing the common project stages and milestones, focusing on the behaviours of the client and the supplier. The description suggests that, due to the root causes outlined in Sections 8.3 to 8.8, the likelihood of poor software products and consequently challenged ISD projects is significant within TSDF.

Stage 1: Client Preparing "Request for Proposal (RFP)"

Consider the case when the client organisation has made a business case and decided to develop an information system with one or more software applications. During Stage 1, the client forms a project team responsible for the initial preparations and considers options available to acquire the required application software, based on which hardware and system software will be decided. When a client decides that an externally developed software application is required, a Request for Proposal (RFP, sometimes called Request for Quotation - RFQ) is prepared to solicit bids from suppliers.

Even at this early stage, the client faces potential PA problems. This happens when the client relies on third party consultants for advice and technical assistance to examine possible options. If the third party consultants are not truly independent, they may influence the RFP in favour of their own companies or business partners. In Case 1, the lead technical architect, whose company is closely affiliated with Microsoft, could have favoured no technical platform other than that from Microsoft (Section 4.3.2). The lack of appreciation of the PA problems can lead to conflicts of interest. For example, Collins (1998) reports that in the RISP Project, Arthur Andersen participated in the eventual bid (jointly with IBM) even though it served as an advisor in feasibility studies and bids evaluation. Such conflicts of interest in ISD projects prevent objective assessment of options and lend unfair competitive advantages to some suppliers. It can thus be detrimental to client's acquiring the best solutions.

Milestone 1: RFP Issued

An RFP may be brief with only a Statement of Work (SOW) or comprehensive with a detailed specification of requirements. In whichever case, the requirements will not be specified "completely, precisely and correctly" (Brooks, 1995; Section 8.3).

Stage 2: Pre-contract Bidding

This stage can be further divided into two. Stage 2a is the period during which competitive bidding takes place. The competition between suppliers is likely to be fierce due to the high switching cost that is not fully observable (Section 8.4) and the client faces an adverse selection problem (Section 8.2).

Stage 2b is from the point a supplier is selected to the point a supplier is contracted. After the selection decision is communicated to the supplier, and before the contract is signed, if the intended contract price is to be of the fixed price type, the supplier has a short opportunity

window to increase the fixed price from the initial P to P':

$$P \leq P' \leq (P + s)$$

In which s refers to the switching cost the client faces in acquiring the software product. The supplier may not go all the way to recover the switching cost at this stage since the client switching cost may be still relatively low. In fact, switching cost is a function of time, increasing as a project progresses. However, even at this stage, by announcing the selected supplier, there is already a considerable switching cost to the client due to financial, political and even psychological reasons (Section 7.3; Section 8.4.1).

Milestone 2: Client Commits To Contract With One Supplier

This milestone is reached when the client gives the supplier a signal to proceed with design and development. The signal may be given as a purchase order or less formally, some instruction from a recognised client authority. A formal contract may or may not be signed subject to legal discussions. However, a legally valid contractual relationship would be established between the client and the supplier nonetheless. Whether signed or not, contracts for substantive software products development are often incomplete (Section 8.5).

Stage 3: Software Development Activities

Software development involves creative activities with actions largely hidden from the client. There are many opportunities for the supplier to make decisions as for what to do and how to it. The rational behaviour for a supplier to adopt is to maximise its utilities. Its behaviours are discussed in the following categories: price, quality, innovation and communication.

Part of the supplier's effort will be to recover the loss built-in as part of the low bidding price and extract as much as possible the return share. Whenever there is an opportunity, the supplier has the incentive to encourage and introduce changes so that monopoly prices can be charged. However, the client may resist changes by "freezing" the project scope or may even

adopt a scope reduction strategy (e.g. Section 4.4). Scope-freezing refers to a situation where a requirement is not added to the project scope despite business needs. The scope reduction goes a step further. The client agrees to reduce scope so that some prioritised part of a software product can be delivered first. A client usually agrees to scope-freezing or scope reduction at the expense of lower business benefits and higher future operational and support costs, in the hope that the project is completed on time and within budget. These are effectively strategies for a client to limit the supplier's monopoly exploitation in the short term. There is also the possibility that the supplier will demand a contract negotiation when the supplier can charge monopoly price for the whole contract. This happened in Case 1 in the shape of "re-planning", where MSS increased its price by 67% (Section 4.5).

When a client successfully adopts a scope-freeze or scope reduction strategy, combined with a strict fixed price contract leaving no room for opportunistic pricing behaviour, the supplier may exploit the monopoly with low quality (Section 8.5). This happened in Case 2 and Case 3 where no price increase was possible after their respective contract effective date. The software product quality was poor in both cases (Section 5.8; Section 6.6).

While the supplier management may wish to keep to a certain level of budget estimate for the product development, the development team requires a trial-and-error approach to find the best solution (Section 8.7). If the conflict between the available resources (in terms of budget and available skills) and the need for trial-and-error is low, the product is more likely to be developed on time with good quality. Otherwise, the opportunity for the development team to innovate is likely to be limited if no sufficient time is allowed for creative experimentation.

Communication from the supplier to the client is likely to be poor at this stage. This was clearly observed in the participant case studies. For example, in Case 2, the supplier AES provided inadequate information regarding the data validation rule (Section 5.9.5). In Case 3,

some of the key resources were simply away working on another project, delaying any possible response to the client's queries (Section 6.8.4). Sometimes, suppliers' management would limit informal contacts between their development teams and their clients' project teams by imposing a single contact point, thus creating a bottleneck for communications. Suppliers may even refuse sharing design documents on the legitimate ground of protecting their intellectual property rights (e.g. Section 6.8.4).

Milestone 3: The First Software Product Release

Part of project team's role is to accept the software product from the supplier and to carry out necessary quality verification including installation, functional and performance testing etc. Therefore, the supplier is required to release the software product a few weeks or even a few months ahead of any information systems implementation deadline. The first release date is a key milestone for the project and for the supplier.

A software product's first release can reveal a lot of what has not been communicated from the supplier to the client up to that point. While by this time, on the client side (the client's ISD project team), resources are mobilised ready to test and install the application, the software release is typically an anticlimax. The common observations (Section 4.4; Section 5.8) are that the first release is often incomplete, does not work and has no proper documentation. The anticlimax was experienced in both Case 1 and Case 2 where the expected releases of software products were first delayed for days and then partially delivered. A partial delivery may be technically meaningless but it buys time for suppliers to improve products to some extent. Consequently, the planned time for integration and quality assurance activities is irrevocably eroded if project deadlines remain the same. Since ISD project teams are also under pressure to complete projects on time and within budget, it is often the case that integration and quality assurance activities are cut back.

Stage 4: Product Evaluation and Integration

Upon receiving the software product, the client carries out acceptance evaluation. The project team installs the product onto a pre-prepared system platform (hardware, system software and the network environment). In addition, the project team tests the product with test datasets. Finally, if applicable, legacy data will be populated to the target database so that the product and the data can be "integrated".

At the initial product release, the supplier tends to promise future improvements. As a result, instead of making any final decision to accept or reject the product, the client enters an iterative process of testing, reporting issues and receiving patches or new releases for testing again (e.g. Section 5.7). Due to the trial-and-error development process (Sections 8.6 and 8.7), the supplier is not able to guarantee the promised improvements. Indeed, a subsequent release may not even be as good as the previous one. One genre of testing called "regression testing" is used to identify "new faults that may have been introduced as current ones are being corrected" (Pfleeger, 1998, p. 338). However, the supplier promises enough so that the client hope is sustained. The client has no practical alternative at this stage.

Milestone 4: Information System Implementation

In reality, the ISD project tends to drift along. Eventually, the client may accept an unsatisfactory software product and its associated system into the operational environment even though software may not be tested and integrated fully with other system elements (especially the legacy data). A continued effort is required from the supplier to improve the software product. Many press reports about ISD project failures are not about cancelled projects before implementation, rather they are about immature systems being implemented causing operational chaos (e.g. LAS CAD system in Flowers, 1996; Tiptree warehousing system in Collins, 1998; Cambridge University' CAPSA Project in Finkelstein, 2001; Greenleaf's OneWorld system in Hayes, 2002).

Premature system implementation causes many problems. To start with, client's operational needs are not met. In Case 2, the client had to rely on alternative workarounds to deal with day-to-day data processing to meet regulatory requirements. LAS CAD Project was a disastrous example of how "lives may well have been lost" (Collins, 1998, p. 166) due to implementation of a premature system. Premature systems leave the client with a system support headache, considerably increasing operational costs. End users' job satisfaction may be seriously affected by having to work with cumbersome systems and having to deal with frequent system exceptions and failures.

Stage 5: Post-Implementation Activities

Activities after implementation can be looked at from different angles. For the client, the system enters the operational stage. Support responsibilities are assigned to appropriate teams responsible for hardware, network, database, application, help desk etc. From the supplier's point of view, development and enhancements continue while acting usually as the second-line support to the client. For the ISD project, there should be a project review. However, project reviews have been conspicuous by its absence in ISD projects. Despite regular calls for post-project reviews (e.g. Whitten, 1995; Cooper et al., 2002; Williams, 2003a, 2003b), they are the exceptions rather than the rule. There are a number of obstacles in acquiring open and unbiased project reviews. First of all, the client may want to keep quiet about a poorly run project. Though few would dispute that more can be learned from failures than from successes, psychologically no one wants to be associated with failures. Unless external pressure is applied, a troubled project is unlikely to offer a detailed review from inside. Secondly, project reviews, when they do take place, may be seriously biased if not managed and co-ordinated by an independent authority. Gulliver (1987) reports that the post-project reviews in BP managed by an "independent unit" contribute to BP's overall financial performance while those performed by project teams might be biased with "preconceived

ideas". However, ISD projects tend to be complex both technically and organisationally, so that post-project reviews by an independent authority may involve considerable costs. Thirdly, controversial review reports are likely to be inaccessible to later project teams, denying organisations learning opportunities. In this respect, both UK and US governments are exceptional. National Audit Office (NAO) and the General Accounting Office (GAO) in the respective governments have audited and published some detailed IT project reviews (e.g.: NAO, 2000, 2003; GAO, 1992a, 1992b, 1994). However, even when a detailed project review does take place and is managed by an independent authority, the complex underlying cause and effect relations may not be immediately obvious for reviewers (Cooper et al, 2002). Williams (2003b) attempts to offer simpler techniques to help practitioners but post-project reviews on a project-by-project basis may not expose root causes common to a group of projects. As a result, the practical value that organisations may gain directly from their own project reviews might be limited. It is therefore not surprising that the reviews often do not take place (Whitten, 1995).

At the private level, however, project team members inevitably ask questions regarding reasons for their project challenges and failures. In relation to suppliers, they may even vow never to use the same suppliers again. However, there are many good reasons that suppliers associated with previously troubled projects are contracted again. New projects may be undertaken by different project teams within the client organisation. Data integration risks mean that incumbent suppliers may be preferred (e.g. Section 5.3). High switching cost forces the client to retain existing suppliers despite possible emotive feelings. The client may simply have no alternative choice. A client may justify such a decision by stating "Better the devil you know" (Bradbury, 2000) while a supplier may rightly feel that it is "no worse than others!" (according to a private conversation with the technical architect from AES in Case 2). The ineffective post-project review process combined with high switching cost might have contributed to the continuing low ISD project success rate.

8.9.2 Modelling TSDF with Game Theory

Section 8.9.1 describes TSDF that can be summarised as follows. Once a client embarks on developing an information system and decides that a software application is to be sourced externally, an RFP is sent out to a number of suppliers. Some suppliers bid for contract and some do not. Only one supplier's bid is accepted. The client offers the winning supplier a contract. The supplier makes the effort to produce the software application. The process can be modelled with concepts and techniques of the game theory (Gardner, 1995; Gibbons, 1992). The game theory is useful to capture some of the essence of business games and help to predict outcomes. The basic concepts and techniques are briefly described below followed by a game theoretic representation for TSDF.

An ISD Project as A Game

Gardner (1995) defines a game as:

> "any rule-governed situation with a well-defined outcome, characterized by strategic interdependence" (p. 4)

ISD projects satisfy the criteria in the above statement. First of all, a client and its suppliers are organisations that are governed by various rules of engagement in business dealings, written or otherwise. Secondly, ISD projects, like other project management endeavours, have finite duration (PMI, 2000) and have outcomes, even though the assessment of the outcomes may not be easy (Bannister et al., 2001). Finally, there is interdependence between all the "players" involved, including the client and all suppliers participated in the contract bidding.

In the discussions below, the "subgame" concept is also used. A subgame is "any part of a game that itself can be played as a game" (Gardner, 1995, p. 148). The subgame concept is useful when the focus of analysis is on part of a game.

Gibbons (1992) groups games into four categories based on whether they are static or

dynamic, and whether game players have complete information or incomplete information (Table 8.3, based on Gibbons, 1992). A game is said to be static if players choose their actions without knowing their opponents' actions. In other words, players act independently. This is normally the case when suppliers bid for a contract. Otherwise, a game is dynamic or sequential. A player is said to have complete information if one has the knowledge of the other players' payoff functions. If the payoff functions are private to the players themselves, the game has incomplete information. Gibbons' classification of the games allows a particular approach to study the games. Another important criterion for examining games is whether a player has perfection information at the decision point. A player has perfect information if all previous moves are known. Otherwise, the game is said to have imperfect information.

Table 8.3: Types of games

	Complete Information	**Incomplete Information**
Static	Static games with complete information, for example: Prisoners' Dilemma	Static games with incomplete information, for example, Sealed Bids in Auctions
Dynamic	Dynamic games with incomplete information, for example, Chess Games	Dynamic games with incomplete information, for example, Open Bids in Auctions

Games may be grouped in other ways, for example, competitive and co-operative games, zero-sum and nonzero-sum games etc. Game theory has been used in studies of many branches of economics and management sciences. Only basic concepts and techniques are used in this thesis. The terms and notations used here mainly follow those of Gardner (1995). It is recognised that much further studies may be carried out to model client-supplier relations using the game theory. These may be pursued as separate research projects.

Game Players' Rationality

The game theory shares the same basic assumption with other economic theories that game players are rational (Williamson, 1985; Arrow, 1986). That is to say that all players try to do their best "in their own eyes" (McMillan, 1992). In the game theory, it is assumed that rational players do not play "a strictly dominated strategy" (Gibbons, 1992, p. 4). Further, it is

assumed that "it is common knowledge that the players are rational" (Gibbons, 1992, p. 7). These assumptions are taken to be idealised conditions under which simpler models can be built for economic analysis. In the real world situations, game players make mistakes. As a result, the concept of "bounded rationality" has been developed to model imperfect behaviours (Gardner, 1995). The following discussions assume that all organisations involved behave rationally. Future research might benefit from more elaborate modelling with more realistic assumptions, including considerations of reputation and long-term interest.

Modelling Client-Supplier Contract Process within TSDF

Software development in TSDF can be modelled as a game in the extensive form (Figure 8.2).

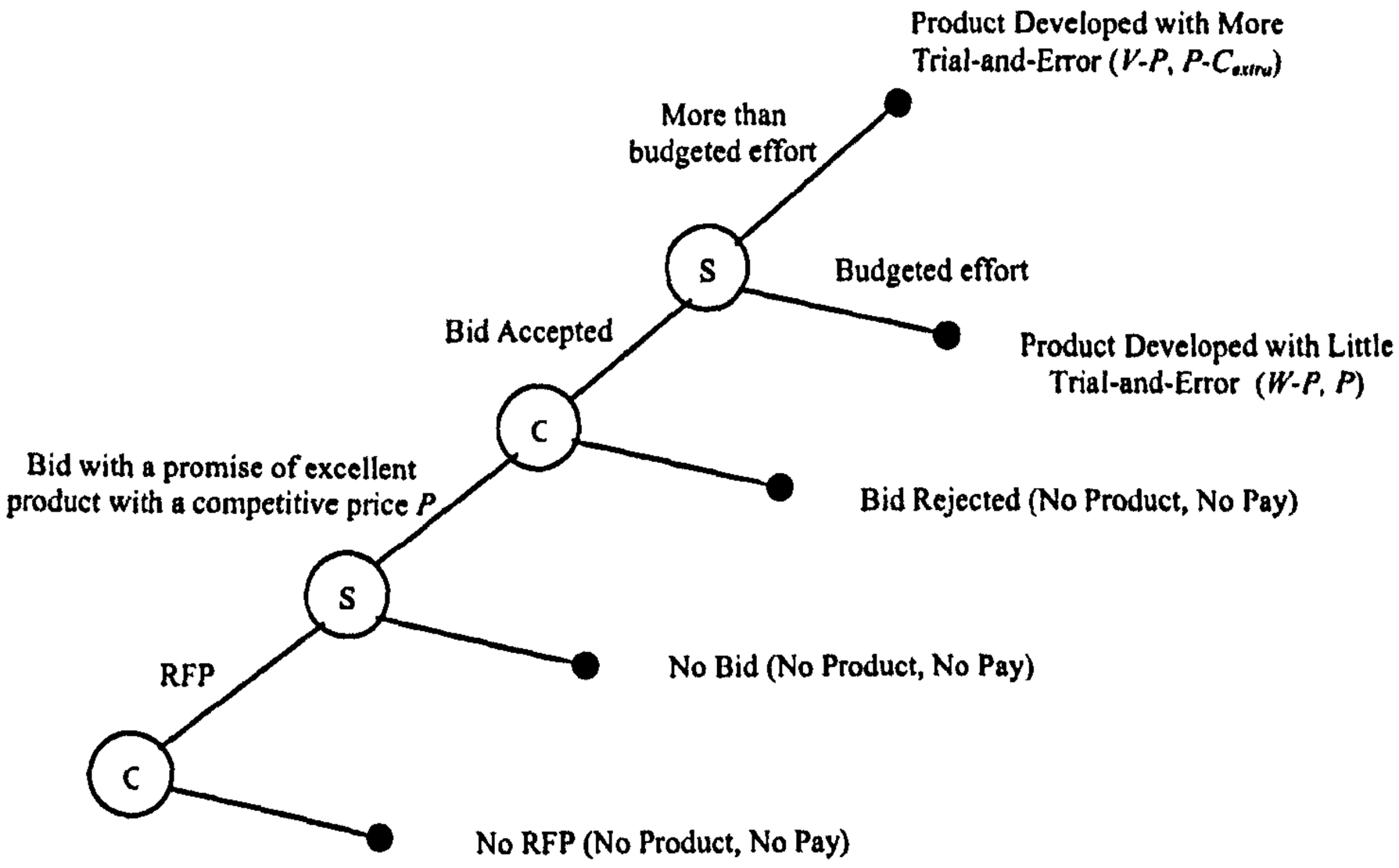


Figure 8.2: The game of software development in TSDF

There are two players in the game: one is the client (C) and the other the software supplier (S). The game starts with the client making a decision on RFP. Deciding not to send out RFP could mean that the client does not plan to proceed with development or intends to develop the software product internally. Sending out RFP is to invite external suppliers to put forward proposals. Next the supplier decides to bid or not. There are many possible actions. In Figure

8.2, the supplier is modelled with either to bid with high promises or not to bid at all. It is possible that a supplier may bid with realistic promises and prices. But in an industry where clients' switching costs are high, such bids are likely to be rejected either for not enough promises or too high prices if other suppliers behave opportunistically. As a result, the supplier bids with a promise of excellent product with a competitive price. Subsequently, the client decides to accept or reject the bid. If the client accepts it, the supplier gets a contract.

The key part of the game is the subgame that is to follow. The supplier gets the contract and exerts effort to develop the promised software product. The budgeted effort usually allows the supplier to deliver a poor product. This is due to two reasons. The first is that the supplier has to cut the budget to the minimum in order to bid with the lowest possible price in the first place. The second reason is that software product development is full of uncertainty. Overcoming the uncertainty requires a trial-and-error approach with a higher level of effort. The budgeted effort is simply not enough to pursue all options to find the best solution. The payoff for the supplier is the bid price P. The client gets a product that is valued at W and pays the supplier P. Its payoff function is therefore $(W\text{-}P)$. W may be so low that $W < P$ (i.e. the product's value to the client is less than the price paid). Alternatively, the supplier may pursue a strategy to exert extra effort above the budgeted level, costing C_{extra} (>0). In that case, the payoff for the supplier is $P\text{-}\ C_{extra}$. The client gets a better product that is valued at V ($V > W$) and still pays the supplier P. Therefore its payoff is $(V\text{-}P)$.

Which of the two strategies would a rational supplier pursue? According to game theory, for the supplier, the strategy of making the budgeted level of effort strictly "dominates" the strategy of making more than budgeted effort due to $P > (P\text{-}C_{extra})$. Since a rational player does not play the strictly dominated strategy, the game theory predicts that the supplier will make the budgeted effort, even though that means the client may be left with a product worth less than the price paid for. The supplier takes this strategy despite its initial promise to

deliver a high quality product. The supplier's initial promise is said to be "non-credible" since making good the promise will not maximise the supplier's own payoff.

8.10 A Critique of Iterative Development Approach

The above sections have explored the possible root causes for the pattern of ISD project challenges. Some of the identified root causes have been recognised in the literature already, in particular the incomplete requirements and the inseparability of design and development. A popular iterative approach has been proposed as a solution. A number of ISDMs have been formulated based on the iterative approach, including SCRUM (Schwaber, 1996), the Rapid Application Development (RAD) methodology (Stapleton, 1997), Rational Unified Process (Royce, 1998; Kruchten, 2000), eXtremeProgramming (XP), (Beck, 2000) and various "agile" methodologies (Highsmith, 2000; Abrahamsson et al., 2002). Of course, the iterative approach is not new and can be traced back to Royce's seminal paper in 1970 (Fitzgerald, 2000). The fact that the existing ISDMs have not been effective to overcome the "software crisis" (Sections 3.6 and 3.7) suggests that there might be fundamental weaknesses with the iterative approach. Some specific ideas and techniques of the iterative approach have already been examined in the discussions of the individual root causes. This Section offers a more systematic critique on the iterative approach including its contributions and possible reasons that the iterative approach may not have helped ISD projects to succeed.

8.10.1 The Iterative Approach and Its Contributions

The iterative approach has been proposed to overcome the weaknesses of the "Waterfall Model" (Royce, 1998; McBreen, 2002a), which is also called the "Systems Development Lifecycle (SDLC)" (Fitzgerald, 2000; Avison and Fitzgerald, 2003) or "the sequential process" (Kruchten, 2000). The iterative approach recognises that requirements cannot be "frozen" (Kruchten, 2000, p. 55). This is an important contribution since it dispels the misconception about the need for complete requirements before design and development

(Section 8.3). The iterative approach also acknowledges the inseparability of design and development (Section 8.6; Kruchten, 2000; McBreen, 2002a). As a result, many ISDMs based on the iterative approach, especially the agile methods, rely on user involvement rather than rigid project planning. At the theoretical level, these ISDMs advocate a flexible, social constructive approach to project management, unlike the traditional planning-based approach (Koskela and Howell, 2002a; Melgrati and Damiani, 2002). Koskela and Howell (2002b) examine SCRUM in particular and concludes that it is based on "alternative theories of planning, execution and control". Although SCRUM and other iterative approach based ISDMs may have taken a step forward to overcome the traditional project management methodology, they have not been successful in overcoming the ISD project challenges (e.g. Sections 3.6, 3.7, 4.6.3 and 4.7.5). Section 8.3 has already discussed the possible weaknesses of the iterative approach to overcome the problem of incomplete requirements. The following subsection focuses on its approach to overcome the inseparability of design and development, i.e. the converging argument. Section 8.10.3 examines how the iterative approach manages the other root causes identified in this Chapter.

8.10.2 The Converging Argument for the Iterative Approach

Highsmith (2000) provides a converging argument for the interactive approach in a diagrammatic form (Figure 8.3, based on Highsmith, 2000, p. 86). The iterative process is depicted as a series of converging steps reaching the desired result.

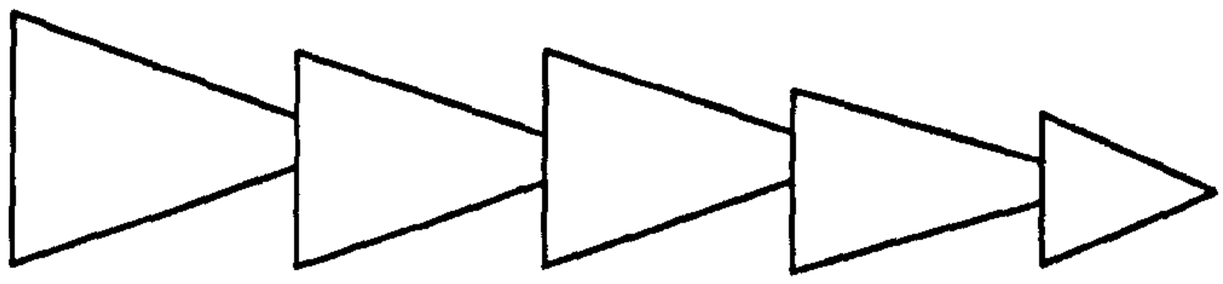


Figure 8.3: The converging argument for the iterative development

However, there are at least two underlying assumptions that must be examined for this argument. One is that the converging path will lead the development to the desired end

product. But this is only possible if the starting position is the "correct" one, which is unknown in the beginning. If the starting position is away from the "correct" place, the result may converge, but will not necessarily converge to the desired end point. The converging process matches the "learning" approach characterised by Pich et al. (2002) in carrying out tasks in projects. The learning approach extends the "instructionist" approach by providing a level of flexibility but it does not accommodate radical changes necessary "within a reasonable number of iterations (Pich et al., 2002, p. 1015; see also Section 8.7.3). Fundamentally, the converging process does not address the problem of development uncertainty (Section 8.7).

The other assumption is that the communication between the client and the supplier is always open and efficient. However, there are usually serious communication barriers between the client and the supplier (e.g. Vitharana and Zahedi, 1997). A supplier may resent "unreasonable" changes requested by a client. A client may fear to be regarded as stupid to demand obvious changes at later stages. Curtis et al. (1988) point out the need to recognise the cognitive processes in user interactions that have not been adequately addressed in the current development methodologies. Beath and Orlikowski (1994) reveal that there are fundamental contradictions in the assigned roles of users and analysts in one ISDM called "Information Engineering" that follows the iterative approach. In the prescribed practices for "Information Engineering", despite the emphasis of user involvement in "joint development", users are portrayed as "naïve, technically unsophisticated, and parochial" while IS analysts are seen as "more knowledgeable, more professional, and more corporate-minded" (Beath and Orlikowski, 1994, p. 372). Such contradictions "are likely to undermine interactions between users and IS analysts" (p. 373). Communication barriers make iterative processes less open and less productive, reducing its effectiveness to tackle incomplete requirements (Section 8.3). Case 1 provides an example to demonstrate how the iterative approach failed to yield a satisfactory result even with a seemingly "excellent" starting point (Section 4.6.3).

8.10.3 Inadequate Considerations for Root Causes

While the iterative approach rightly acknowledges the problem of incomplete requirements and inseparability of design and development, it adopts the client-supplier partnership philosophy to manage the client-supplier relationship (e.g. Highsmith, 2000). As a result, ISDMs based on the iterative approach do not deal with possible supplier opportunistic behaviours. Therefore, they do not discuss the challenges caused by high switching cost (Section 8.4) and incomplete contracts (Section 8.5). Instead, other techniques like "timeboxing" are used to manage project cost and schedule. In addition, the concept of "good enough software" has been proposed to manage software quality. Both are examined below.

Timeboxing

The timeboxing technique is used by Rapid Application Development (RAD), later standardised as the Dynamic Systems Development Method (DSDM, undated-a). It is based on the observation that in the sequential process, the functionality of a software product is supposed to be fixed while schedule and resources are varied in a project to achieve the fixed functionality. Therefore the proponents of the timeboxing technique suggest varying functionality within a fixed timescale and a fixed level of resources called "timeboxes". The timeboxing technique forces a software product to be produced however limited its functions might be (Figure 8.4, from DSDM, undated-b; Helpsoft, undated, see also Avison and Fitzgerald, 2003, pp. 95-97).

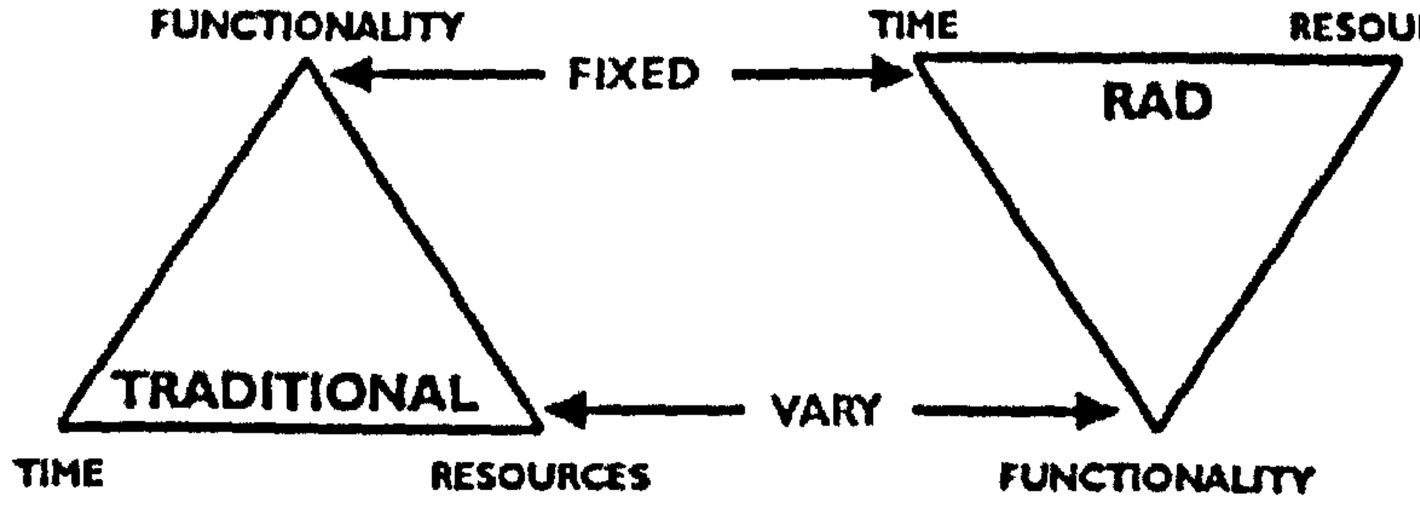


Figure 8.4: Fixing time and resources in "timeboxes" and varying functionality

The timeboxing technique assumes that after a series of timeboxes, the developed software product will largely meet the client requirements though "some of the things that the system was going to do will be jettisoned" (Avison and Fitzgerald, 2003, p. 97). The technique does not and cannot provide answers to questions like "what percentage and which part of the system will be left undeveloped". Though the technique recognises the problem of development uncertainty (Section 8.7), it does not deal with it sufficiently to meet client requirements. It also fails to recognise the imperfect communication process that may lead the converging process to unsatisfactory end products (Section 8.10.2). The timeboxing technique was used to develop the front-end WebShop pages in Case 1. It was true that software artefacts were delivered to the client all the time, however imperfect they might have been. But when the project ran out of the "timeboxes", the supplier NQ refused to carry out further work. In this particular case, the supplier delivered the design within the budget limit and on schedule. However, the client was "dissatisfied" with the deliverables (Section 4.7.5).

The "Good Enough" Software

A natural question to pose about the iterative process is when the iteration should stop. The concept of "good enough software" emerged in 1990s (Bach, 1995, 1997; Yourdon, 1995; Stapleton, 1997) to answer this question. The concept describes a software development strategy that the "right" defects might be left in software products delivered to clients. It is essentially an economic argument that software defects should be fixed based on cost-benefit analysis, or in terms of return for investment:

> "If something is really good enough, ... then further improvement means making an investment that has an inadequate return." (Bach, 1997)

The concept is distilled from real world projects and is not necessary new (Bach, 1997). But proposing it as a criterion for software evaluation in ISD projects has been controversial. Should it be the client or the supplier who decides whether a software product is "good

enough"? What further criteria should be used to judge whether the "good enough" point is reached? Yourdon (1995) suggests "a rational negotiation" between the client and the supplier to agree "*all* of the relevant success criteria" (emphasis in the original). However, Yourdon admits that the relationships between the project constraints and level of quality are not well understood and it will be "difficult" for clients to agree to a particular level of defects. This is in fact the problem of absolute product evaluation (Section 8.8). Therefore, the concept of "good enough" software is essentially theoretical. Its actual implementation is highly problematic (see e.g. McBreen, 2002b).

In summary, the iterative approach recognises two fundamental root causes of ISD project challenges, namely incomplete requirements, the inseparability of design and development. However, it does not consider the complex principal-agent relationship between the client and the supplier, thus ignoring suppliers' opportunistic behaviours resulted from high switching cost and incomplete contracts. The iterative approach assumes an open and efficient client-supplier communication process, which in fact is full of barriers. The converging argument underlying the iterative approach does not facilitate innovative alternative solutions and therefore does not resolve the fundamental problem of development uncertainty. In addition, it does not resolve the problem of absolute product evaluation. It is for these reasons that the iterative approach might have failed to overcome many of the ISD project challenges.

8.11 A Unified Explanation

Sections 8.3 to 8.8 have discussed six root causes as explanations to the pattern of ISD project challenges. Section 8.9 further synthesizes the root causes and suggests that ISD project challenges result from TSDF characterised by competitive bidding and monopolised development. Section 8.10 provides a more comprehensive critique to the popular iterative software development approach. This Section proposes a unified explanation as a basis for formulating a possible solution to overcome the identified root causes:

A Unified Explanation: The fundamental source of ISD project challenges is TSDF characterised by a process of competitive-bidding-monopolised-development. This process allows suppliers to adopt bargains-then-ripoffs opportunistic behaviour pattern due to complex interactions of the root causes including high switching cost, incomplete requirements, incomplete contracts, inseparability of design and development, development uncertainty, and client's difficulties in devising and enforcing acceptance criteria. Though iterative-based ISDMs have recognised the problems of incomplete requirements and inseparability of design and development, they have not acknowledged the complex PA relationship between a client and a supplier. In addition, the learning approach underlying the iterative ISDMs does not resolve development uncertainty. As a result, they have not been able to resolve the fundamental weaknesses associated with TSDF.

The above explanation implies a solution with competitive development and a selectionist approach. A competitive product development process entails two or more delivered products for the client to evaluate, thus overcoming the problem of absolute product evaluation. Two or more parallel product development efforts allow two or more promising solutions to be explored, thus at least partially overcoming the problem of development uncertainty.

The unified explanation can explain the pattern of ISD project challenges as analysed in this chapter. It is both consilient and simple. These are two of Thagard's (1978) central criteria of good explanations (Section 2.6.2). However, a number of questions remain, for example:

- Is this a better explanation compared with other ones?
- How feasible is the implied solution for software development?
- How could the high switching cost be controlled so that it does not distort suppliers' bidding pattern?

- How can the selectionist approach be implemented?

The unified explanation is effectively a hypothesis, which cannot be easily falsified or validated as it stands. Attempts are made in Chapter 9 to answer the first two questions. This is done by following the "Inference to the Best Explanation (IBE)" reasoning process (Section 2.6.3). Two contrastive analyses are conducted to test the validity of the explanation followed by two analogical analyses exploring its feasibility on the solutions implied by the explanation. In the long run, action research studies might be carried out by introducing changes to real world ISD projects based on this hypothesis (see implications for future studies in Section 11.5). A management framework based on the above explanation is suggested in Chapter 10 for this purpose.

8.12 Summary

This chapter attempts to identify a best potential explanation for the pattern of ISD project challenges recorded in Chapter 7 by following the IBE reasoning strategy. First of all, it is suggested that the client-supplier relationship is examined as that of principal and agent rather than that of partnership. The principal and the agent are assumed to be economically rational when entering a commercial contract. An effective contract must meet the incentive compatibility constraint and the participant constraint. In order to explain the challenges in ISD projects, the basic PA framework is supplemented by the concepts of switching cost, incomplete contracts and product evaluation methods.

The steps of supplier selection, contracting, contract execution and product evaluation are traced to see how ISD project challenges can emerge and develop within TSDF. High switching cost is a key feature of the software market. Coupled with incomplete requirements, incomplete contracts, inseparability of design and development, development uncertainty, and inherent difficulties in evaluating software products and diagnosing software defects, suppliers adopt a bargains-then-ripoffs strategy. The consequence is that suppliers promise

excellent quality at low prices and deliver poor quality at higher prices. The client-supplier contracting process may be modelled as a game with the supplier giving "non-credible" promises. To fulfil the supplier's usually high promises at the bidding stage, the supplier has to make extra effort due to the inherent nature of software development calling for trial-and-error approach. A rational supplier chooses a strategy that maximizes its own payoff even though such a strategy results in a software product worth less than its price to the client.

A critique is made of the popular iterative approach. Though recognising the problems of incomplete requirements and inseparability of design and development, the iterative-based ISDMs adopt a learning approach which does not effectively address development uncertainty. They have also assumed client-supplier relationship to be that of partnership, therefore have not acknowledged the complex PA relationship between a client and a supplier. As a result, the iterative approach has not been able to resolve the fundamental weaknesses associated with TSDF.

The chapter ends with a best potential explanation (the unified explanation) that is consilient and simple. The process of suggesting this best potential explanation has been laid bare in this chapter for readers to scrutinise, and therefore, the explanation is more than just a pure conjecture in the Popperian sense (Popper, 1963; Section 2.4.1). However, the explanation remains as the best potential explanation similar to a hypothesis without going through a set of contrastive analyses or "causal triangulation" (Lipton, 1991; Section 2.6.3). Chapter 9 provides two contrastive analyses to test the validity of the unified explanation and two analogical analyses to explore the feasibility of the solutions implied.

Chapter 9: Comparing ISD Projects with Other Types of Product Development

There can be no differentiation without contrast.
(Chinese Proverb)

9.1 Introduction

Chapter 8 explains the observed pattern of ISD project challenges by exploring a number of possible root causes. It ends with a unified explanation, suggesting the underlying cause is TSDF characterised by competitive-bidding and monopolised-development. The implied solution is therefore to replace the monopolised development with a competitive mechanism. To examine the validity of the explanation, software development in ISD projects is contrasted with that in PPSW market segment, where competitive force is at full play (Section 9.2). The construction industry is examined next (Section 9.3), anticipating a potential counter-argument that monopolised product development can be successful. The two contrastive analyses suggest that the unified explanation given in Chapter 8 might be considered as a "warranted inference". To examine the feasibility of the implied competitive development solution, an analogical study is conducted comparing ISD projects with the "black-box" component design process in the manufacturing industry in Section 9.4. In addition, the Open Source Software (OSS) development process is examined in Section 9.5 to see the development competition at work. Section 9.6 summarises the chapter.

9.2 Contrasting PPSW Market with ISD Projects

Researchers and practitioners often attribute software development failures to the immaturity of the software industry or the complexity of software products (Brooks, 1995). In other words, "software is different" (Beizer, 2001). It is true that the software industry has only had fifty years of history. It is also true that software products can be very complex. However, is

this a good explanation for the pattern of ISD project challenges? If it is, it would imply that software development in general would be not be successful. This is not supported by the empirical evidence from the PPSW market segment.

9.2.1 PPSW – Software Development Can Be Successful

The report from Nathan Associates (1997) describes the software industry as "a powerful engine for US economic growth" and ranks the software industry alone as the third largest in US. A separate PWC (1999) report holds the same view of the software industry in the global economies by stating:

> "The software industry is a remarkable engine of economic growth around the world and holds tremendous potential for the future. In the past several decades, this industry has grown from infancy to robust maturity, becoming one of the most significant sectors of the global economy, with spillover benefits extending to virtually every corner of the world economy."

While we have been concerned with the "software crisis", PWC's report refers to the software industry as being in a status of "robust maturity"! When we look at the report closely, however, it is apparent that the successes are really only attributed to the PPSW market segment. Nathan Associates (1997) only looks at "the packaged business software segment". On the other hand, comments made about the failures of software industry usually refer to ISD projects involving CBSW or COTS based customisation and system integration. The evidence points to a dichotomy of the software market with PPSW products being remarkably successful and COTS and CBSW products used in ISD projects otherwise. Such a dichotomy may explain the conflicting impressions from different sources regarding the software successes and failures as experienced by Glass (1999). Rather than branding the failures statistics as "relatively worthless" (Glass, 1999, p. 2), the success stories and the failure statistics may have originated from two different parts of the software industry.

That PPSW and CBSW are two distinctively different market segments is further confirmed by their historical price index profiles. Parker and Grimm (2000) provide "quality adjusted"

software price indexes for three software categories (PPSW, CBSW and in-house or "own-account" development) from 1959 to 1998. The idea of quality-adjusted pricing, also called "hedonic pricing", is summarised by Hollanders and Meijers (2001):

> "...it assumes that each product is made up of a multitude of definable characteristics, that for each characteristic a price can be estimated and that quality changes in a product can be viewed as adding a new characteristic to the product. The resulting price change can then be divided between the change resulting from adding the better quality characteristic and from a more general price increase (or decrease). As such, a quality-adjusted or 'pure' price can be calculated."

Four price indexes are shown in Figure 9.1 (adapted from Hollanders and Meijers, 2001, in turn based on Parker and Grimm, 2000). The index for PPSW ("pre-packaged" in Figure 9.1) shows that it has fallen sharply. On the other hand, prices for internal software development (both "business own-account" and "government own-account" in Figure 9.1) have increased steadily. As for CBSW ("custom" in Figure 9.1), it shows a trend of steady increase since late 1960s. One conclusion to be drawn from Figure 9.1 is that PPSW and CBSW are two distinctive segments in the same software market. PPSW segment has been able to improve on product quality and reduce product prices dramatically while CBSW together with the internal development segment has failed to do so.

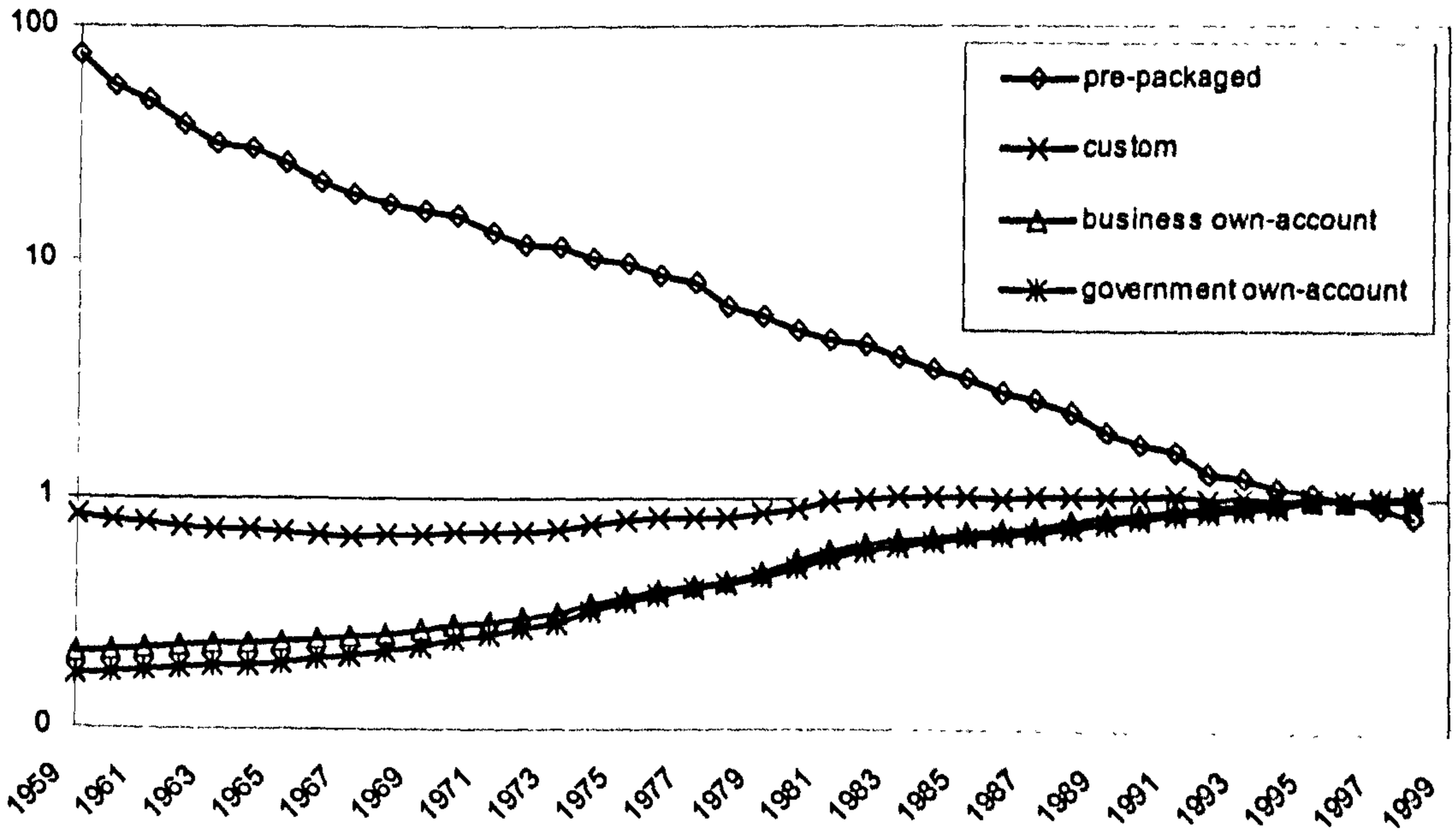


Figure 9.1: Quality-adjusted price indexes for PPSW, CBSW and internal software development, with the 1996 price index set as 1.0

Such contrasting achievements are not explained either by software technology immaturity or software complexity since both are free to draw from the same technology pool. So far as software complexity is concerned, familiar PPSW products like popular operating systems (e.g. Unix and Windows series products), commercial databases, word processing and spreadsheet packages are most complex of all software products. The complexity comes from not only the sizes of the applications themselves, but also from the fact that PPSW products have to be developed to be compatible with many platforms. Brooks (1975, 1995) conjectures that developing a general-purpose application takes nine times extra work than the equivalent for a single customised installation. However, even with recognition of the extra complexity and cost, Brooks (1995) urges people to buy rather than build. He states:

> "From the discipline's viewpoint, the mass-market software is almost a new industry compared to that of the development of custom software, whether in-house or out-house." (p. 218)

Brooks clearly recognises that PPSW and CBSW are two distinctive industries. However, Brooks' advice is difficult to implement since there is often no mass-market software available to meet specific organisational needs.

Has COTS been historically successful like PPSW or less so like CBSW? There is no separate category for COTS in Figure 9.1. However, the original definition of CBSW in Parker and Grimm (2000) includes "new computer programs as well as programs incorporating preexisting or standardized modules", indicating that the remaining development, customisation and integration activities to turn COTS into information systems are treated more like CBSW. The difference of PPSW, COTS and CBSW may be illustrated by the level of pre-contract and post-contract development activities *before a client can make use of the acquired software products* (Figure 9.2).

As shown in Figure 9.2, a supplier completes design and development before supplying

PPSW to clients. Apart from the effort for installation, there is no further development to be carried out. It is exactly the opposite with CBSW, which is designed and developed after a contract is signed with a client. In addition, there is business process and data integration and other activities to be carried out before CBSW can be incorporated fully into an information system for use. The situation with COTS is in between PPSW and CBSW. The exact proportion of pre-contract development and post-contract activities varies with each COTS product. The post-contract activities include customisation, configuration, business process and data integration etc. These activities may also be broadly called as development since they require technical resources, often from COTS suppliers or supplier designated specialists (see e.g. Section 6.3 about system integration). Post-contract development activities using COTS introduce unique issues and challenges (Langley, 1994; Brownsword et al., 1998). Jarzombek (1998) refers COTS as a "double-edged sword". Even implementing relative mature COTS software packages can be risky. This has been demonstrated by Case 2 (Chapter 5), Case 3 (Chapter 6) and many Enterprise Resource Planning (ERP) projects (e.g. Girard and Farmer, 1999; Wheatley, 2000; Katz, 2001). Therefore, COTS may be considered in the category of CBSW in terms of its historical success, or the lack of it.

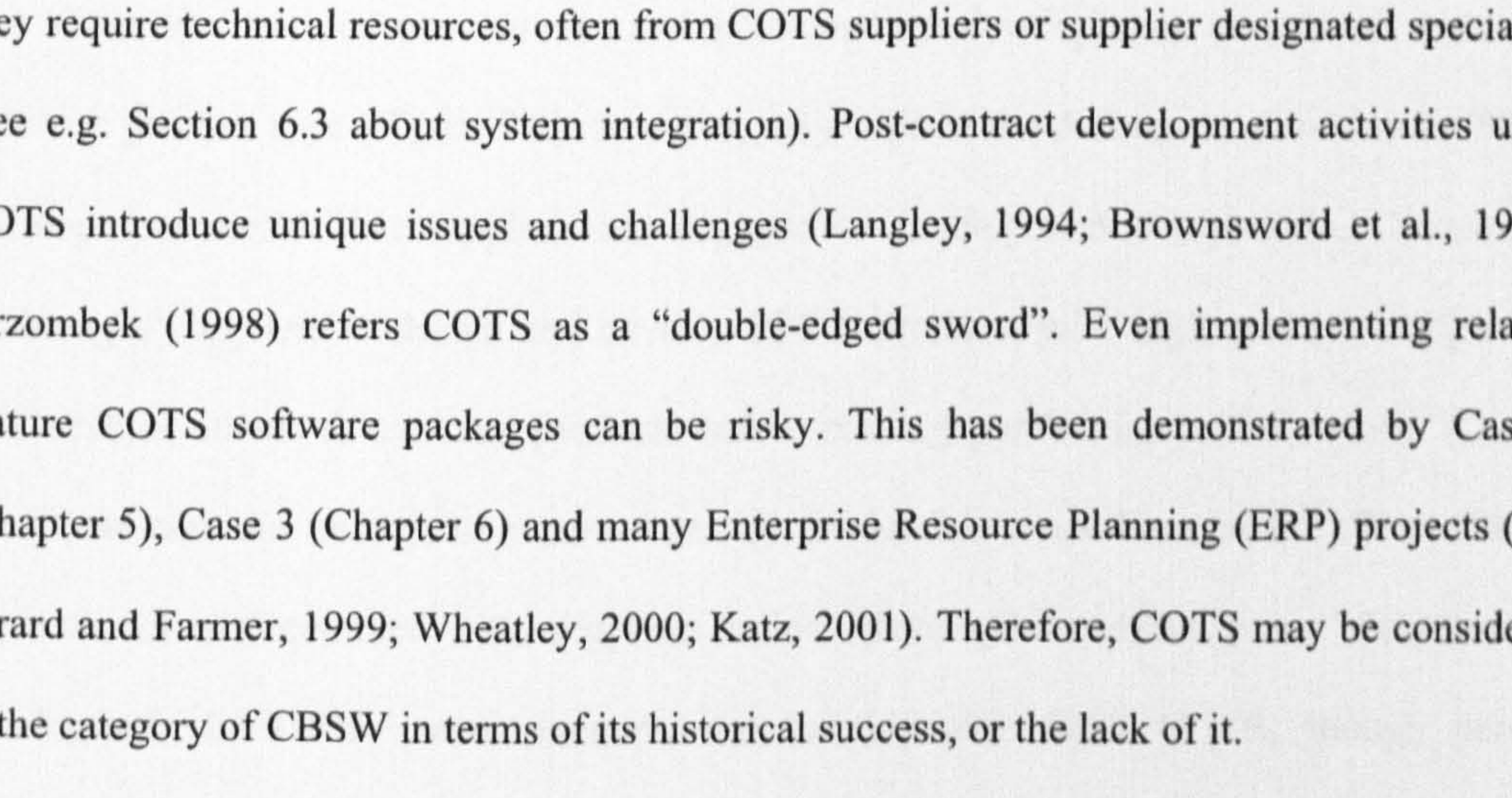


Figure 9.2: Comparing PPSW, COTS and CBSW in respect of pre-Contract and post-contract development and other activities before providing client with benefits.

9.2.2 Different Market Mechanisms and Different Development Approaches

The above discussion demonstrates that PPSW products have been highly successful in the market while COTS/CBSW have been less so. Since PPSW, COTS and CBSW all draw from the same technology pool, the argument that software technology is immature cannot explain the difference. Software complexity cannot explain the difference either since PPSW is potentially more complex due to the need to cater for different platforms.

The different level of pre-contract and post-contract development associated with each software product type (Figure 9.2), however, may explain the difference. Pre-contract PPSW development means that suppliers are engaged in development competition. Competition encourages suppliers to design and develop PPSW products with high quality. Competition also enables clients to evaluate products before making purchasing commitments. Whether clients are cautious enough to make a thorough evaluation is another matter[28]. With CBSW, however, clients contract with suppliers before development takes place. Post-contract development in TSDF means monopolised development. With COTS, though part of development has been undertaken prior to client contracting, the client bears considerable risks with the post-contract activities necessary to develop finished information systems.

The different market mechanisms for PPSW and CBSW may influence profoundly the ways products are developed. There have been several studies contrasting PPSW with CBSW development practices (Carmel and Sawyer, 1998; Sawyer, 2000). The observations are that they are developed with different cultural milieus, different types of teams and different incentives for developers. First of all, organisations for PPSW have entrepreneurial and

[28] Corporate and home users of software products tend to take different approaches to assess software products. For corporate users, user groups can be an important source of information. Actual trials with limited financial commitment are possible. For home users, product reviews in popular trade magazines are often useful. However, in both cases, it is difficult to experience software products fully due to licensing constraints (see Kaner & Pels, 1998), time restriction or unrealistic test data.

individualistic cultural "milieus". In contrast, CBSW organisations are bureaucratic. Secondly, PPSW teams share the same product vision while CBSW teams rely on formal specifications and documents. Thirdly, developers in PPSW teams can expect large financial rewards and more motivated while those in CBSW teams are typically salary-based. Though the role of users is increasingly seen as critical to successful software development, users are generally not integral to PPSW development efforts. Another unusual observation is that PPSW teams' processes are "immature" compared to CBSW teams.

The last two observations challenge some of the received wisdom in the current thinking on software development methodology. "User involvement" is seen to be a key to successful ISD projects (Bronzite, 2000). Yet PPSW development is largely detached from user involvement. In addition, according to Capability Maturity Model (CMM), process maturity or quality determines software product success (Olson et al., 1993; Paulk, 1997). Yet CBSW segment with "more mature processes" is not as successful as PPSW. These contradictions, however, might be explained by the competitive mechanism in PPSW market and its absence in CBSW development. PPSW suppliers, being in a competitive market from design, development to sales, develop products meeting the users' "expected requirements" as well as "revealed requirements" defined in Kano Model (Section 8.3). To win clients in the market place, PPSW suppliers design in delightful features as well. Even though suppliers seem to be detached from the users, the developers think on behalf of users in terms of cost, reliability and usability etc. The competitive "invisible hand" is operating not only on the price dimension, but also on functions, features and their quality dimensions. CBSW suppliers, however, may aim merely for meeting revealed requirements. Any extra feature either has to be paid for as extra, or is strictly controlled by suppliers to reduce development cost. Competition, or the lack of it, therefore, affects profoundly the outcomes of the different product development processes.

Of course, even with fierce competition, PPSW products are not always delivered to the market on time with promised quality, perhaps due to development uncertainty (Section 8.7). In such a case, it is the suppliers who suffer from the results. Clients are shielded from any potential negative impact since they usually do not make purchasing commitments until after products are available. In the case of CBSW, the clients contract with suppliers with pre-defined time and cost commitments. In addition, the clients plan other dependent activities (e.g. data migration, legacy system switch-off, new system support arrangement, new working practices associated with the new system etc) based on project plans before software products are developed. Such close dependency on the outcome of CBSW development causes a chain reaction from late or failed software delivery to late or failed ISD projects, and to potential chaos in the clients' organisations.

In summary, PPSW successes indicate that the software industry has the know-how to build complex and successful software applications. A key difference between PPSW and COTS/CBSW is whether product development takes place before or after a client contracts with a supplier. Pre-contract development for PPSW means development competition while post-contract development in TSDF means development monopoly. Using COTS in ISD projects involves considerable uncertainty with COTS products customisation and integration, demanding suppliers' resources. The different market mechanisms causes software products to be developed in different ways, resulting in different product performances in the market place in terms of functionality and quality. Of course, PPSW are not guaranteed to be successful during the pre-contract development process. When PPSW suffers from development delays and quality problems, clients are protected through the product-first-contract-later mechanism. Such a competitive mechanism forces PPSW suppliers to be innovative in designing and developing products. On the contrary, TSDF means contract-first-development-later mechanism that induces suppliers to be less entrepreneurial. In addition, clients' organisations tend to be closely coupled with bespoke ISD projects with high

dependency on timely delivery of quality CBSW/COTS products. When the expected products fail to be delivered, ISD projects and in many cases, the dependent clients' organisations are thrown into chaos.

While the contrast between PPSW and CBSW/COTS products supports the explanation that the lack of competition in TSDF may be the cause for the pattern of ISD project challenges, it is important to acknowledge that monopolised development may have worked well elsewhere. A particular good example is the construction industry. It may be argued that the competitive-bidding-monopolised-development management framework has been successful in construction projects. The next section examines whether the construction metaphor is an appropriate one for ISD projects.

9.3 Construction Projects: An Appropriate Metaphor?

Software development is a young industry. Naturally, practitioners adopt terms and methods from more mature industries. One of the most well established industries is that of construction and civil engineering. In a well-received book by the software development community, McConnell (1993) systematically explains many "best practices" by using construction as a metaphor. Despite such efforts, failures of software development contrast sharply with the successes of construction projects. Standish Group (1995) laments about software development being not as successful as "bridge building". However, the analysis in this Section demonstrates that software products have different characteristics from construction products so that the construction metaphor is a misleading one. In Section 9.3.1, the characteristics of construction projects are outlined in contrast with those of ISD projects. Section 9.3.2 provides a critique regarding the validity of the construction metaphor.

9.3.1 Contrasting Construction and ISD projects

ISD projects are deceptively like construction projects. Both go through similar phases like

requirements specification, design, development and acceptance etc. Both types of products often call for unique product designs and have to overcome unique engineering problems. Mass production techniques are not easily applicable for reducing costs. Also, the cost of a construction product is usually established by either negotiation or competitive tendering prior to production (Kwakye, 1997). The same goes for software projects. The metaphor of the construction project is so entrenched in the software industry that terms like "software architects" and "software construction" are used without hesitation. However, the two types of projects have fundamentally different characteristics. This is related to the different products they aim to produce. While a construction project aims to produce a *physical* construction product, an ISD project aims to produce an information system, usually based on a core software application, which is a *virtual* product. This fundamental difference determines that construction projects can be managed in ways that cannot be easily duplicated for ISD projects. Notable characteristics of construction projects are discussed below in contrast with ISD projects, including separate design and development, flexible and efficient subcontracting labour usage and real-time independent quality and cost monitoring.

First of all, construction projects can have separate design and build stages (Kwakye, 1997). Importantly, due to the physical nature of the construction products, design can be detailed and controlled with drawings and specification. The separation of design and build allows the design to pass the design quality control first. It is after a design is approved when suppliers start bidding for development (building). Thanks to the availability of the detailed design, contracting can be relatively complete and production quality control can be meaningfully enforced. An architectural design embodies a client's requirements and is a detailed and complete blueprint for building. This does not imply no variations in the process of construction. It does mean that neither the client nor the builder can easily introduce variations without being noticed. Therefore, any scope or quality deviation from the design can be more easily identified and dealt with. The same cannot be said for software

development. As Standish Group (1995) comments:

> "One of the biggest reasons bridges come in on-time, on-budget and do not fall down is because of the extreme detail of design. The design is frozen and the contractor has little flexibility in changing the specifications."

The lack of detailed design forces ISD projects to adopt a contractual framework to combine design and development to be carried out by the same supplier based on incomplete requirements (Section 8.6). The separation of software design and development is so difficult that the whole software development process has been referred to as "pure design" (Callahan and Moretton, 1998; Eischen, 2002). A design might be defined as "a plan for an artefact or a system of artefacts" (Gorb, 1990, quoted in Reece, 1999). In this sense, a design of a software application can be said to be a plan to integrate client requirements and the necessary technical tools and facilities to become specific system artefacts. Of course, software is not merely a design if "a plan" is interpreted as only ideas on paper. Software design, if it is of any use, has to be implemented to operate with data and business processes. Whether a software design actually works or not is not easily determined by looking at it on paper. It has to be proven by implementation, typically through a trial-and-error approach. The inseparability of design and development necessitates a trial-and-error approach to effectively work out design and implementation at the same time (Section 8.6), unless the software is trivial or exactly the same as some software artefacts developed before.

The second observation is related to the efficiency of sub-contracting labour usage. In a construction project, subcontracted labours like plumbers and electricians are called for when required. If the pre-planned work is not ready to be performed, the subcontractors may be told to come back another day. If that does not suit a specific subcontractor, another may be arranged to do the work instead. The client does not normally pay for any subcontractor's waiting time. Thus there is a high degree of labour flexibility in planning and executing building tasks (e.g. Underhill, 2002). The same cannot be said about software projects. Software project team members both in the software development stage and the system

integration stage require a period of familiarisation with a project since every project is knowledge-intensive, or more precisely domain-knowledge-intensive (Eischen, 2002), requiring the understanding of the context of different tasks and activities. This causes a degree of inflexibility or "stickiness" in labour usage. For example, during system integration stage in an ISD project, there are typically resources planned for software installation and testing upon receiving a software product from a supplier. Once the resources have spent their time acquiring knowledge about the project (the business process, test data and technical environment etc), they remain in their positions even when the software product is delivered late by a number of days or weeks. To release these resources temporarily is risky for the project since they may be assigned to other projects. Therefore, the project pays for resources' waiting time. The bigger a project is, the more waiting a product delivery delay is likely to cause, and the higher the cost of waiting. There are two possible ways to deal with such lost time. One is to delay the whole project so subsequent tasks can be performed as planned. This delays the project and increases the cost. The second option is to cut back the planned tasks or reduce effort to save time. This will result in lowered product quality. Often, it is a combination of project delay, cost increase and poorer product quality (e.g. Section 4.4).

Another important process feature of a construction project is that production cost and quality monitoring is independent and real-time. A construction project organisation has a clear line of demarcation during the production phase. While the contractor (the supplier) is responsible for production, the client usually appoints an independent supervision team including the architect, "clerk of works", structural engineer, services engineer and quantity surveyor (Kwakye, 1997). Most notably, there is usually one quantity surveyor working for the client and one working for the contractor. Both quantity surveyors have a duty to monitor project costs. Kwakye (1997) explains the need for two quantity surveyors in the same project:

> "...cost monitoring is of great concern to both quantity surveyors, it is for different objectives. From the view of the contractor's quantity surveyor, his or her concern centres around the total value of the project and the total cost of

> resources in use and sub-contractors. The client's quantity surveyor, on the other hand, considers the payment the client makes to the contractor for executing the construction project and reports immediately to the architect whenever this payment is likely to exceed the client's cost limit. The cost monitoring exercises keep both the client and the contractor abreast of their respective current financial positions. It also enable them to take prompt corrective measures where the need arises." (p. 199)

Such a cost monitoring arrangement deserves a close look. First of all, the independent cost monitoring becomes possible due to the cost structure of a construction project, which includes land, materials, capital and labour. The inputs are turned into visible, physical outputs that can be verified independently and progressively. This contrasts with subcontracting a software product development, where most of the cost is labour, and the output is not easily verifiable independently until the end (Royce, 1970). Secondly, costs are closely monitored by *two* quantity surveyors, not one. Why is there the need for two quantity surveyors? Could one surveyor not serve the whole project so to save the cost for the other? Or if the workload is so high, could two surveyors not work for the contractor or the client at the same time? The construction industry appears to have recognised a profound PA problem, which is the agent's information may not always be presented in the best interest of the principal, and vice versa. To have two quantity surveyors appears to be economically more efficient than to have only one. Thirdly, the cost information is prepared by trained cost professionals. A quantity surveyor's career path starts from a degree course with further two-year work-based structured training programme (Hobsons, 2002). There is no such equivalent cost specialist in ISD projects.

These characteristics of construction projects, namely, the physical nature of the products, the separability of design and development, the flexibility and efficiency of labour resource usage and the practice of feasible independent cost monitoring are in sharp contrast to those of ISD projects. With these fundamental product and process differences, the validity of the construction metaphor for software development might be questioned.

9.3.2 A Critique of the Construction Metaphor

There have been critics to the construction metaphor for ISD projects in the literature. Royce (2002) comments:

> "It's tempting to compare software construction to the building of a house. After all, both projects involve requirements management, design (blueprints), scheduling, speciality teams (comparable to roofers, carpenters, plumbers, etc.), and inspections.
>
> But decades of software projects have shown us that traditional modes of construction are very different from the diverse ways in which software is designed and delivered to the customer. One reason for such a low success rate is that traditional project-management approaches do not account for the level of creativity often required to complete software projects..."

Royce clearly believes that the key difference is the "level of creativity" required for software development. That is valid since in construction projects, contracts with suppliers are for building work, not for design. The main focus for a contract is for a supplier to conform to an approved design though some level of creativity is required even in doing that. In ISD projects, a software supplier is contracted to perform an integral design and development task due to the inseparable nature of the two activities. Such an integral and uncertain task would indeed demand a much higher level of creativity on the supplier side.

However, there is more than the question of creativity. Due to the lack of an approved design in the initial stage of an ISD project, there is not even the basis to form a conventional contract between a client and a software supplier. In TSDF, client requirements have been used as the basis for supplier contract. Yet, as has been elaborated earlier, client requirements will always be incomplete according to the Kano Model (Section 8.3). What is missing is a separate design stage to produce an approved design. Such a design must be detailed for cost estimation, for development planning and execution and for quality verification. The closest output resembling such a design in an ISD project is a "Technical Specification". Unfortunately, the technical specification is typically produced by the software supplier after being contracted. Therefore, it cannot be used as the basis of supplier contracting itself.

There are other significant differences between the traditional projects and ISD projects. Roetzheim (1993) suggests a number of differences at project task level during project planning (including differences in task decomposition, task dependencies, resource requirements estimates and risk analysis etc) and project tracking and controlling. Maxman (2003), along similar lines to Roetzheim (1993), acknowledges that software development projects are more like research and development and "really are different from 'traditional' project management, i.e. projects involving well-established technology".

It is worthwhile to note that there is a "non-conventional" type of procurement system called "design-build" introduced to the construction industry in recent years (Kwakye, 1997; Levy, 2002). In contrast, the conventional construction procurement system might be called the "design-bid-build" process (Levy, 2002). The introduction of this new approach has been controversial and there are on-going debates regarding its pros and cons (Rowlinson, 1988; Rowlinson and Lee, 2000; Chan and Chan, 2000; Levy, 2002). The design-build approach is different from the traditional one in that design and build are combined and managed by the same contractor. In other words, it adopts the procurement framework that, according to the analysis in this thesis, is one of the root causes for ISD project challenges (Section 8.6). According to its advocates in the construction industry, this approach apparently increases production efficiency due to overlapping design and development, among others (Rowlinson, 1988; Kwakye, 1997). At the same time, there are serious observed drawbacks, two of which are:

- "The client may become stranded with a construction product which is unsuitable for his or her needs.
- The client will find it difficult and/or costly to introduce variations once product has commenced on site." (Kwakye, 1997, p. 299)

It has to be questioned that if a client gets a construction product "unsuitable for his or her needs", whether any production efficiency is desirable at all! The difficulty to introduce variations to product design reduces flexibility of the contracting arrangement and increases the likelihood that the end product may not be fit for the client. These drawbacks are strikingly similar to the issues found in ISD projects. Unfortunately, there are "confusions of definition and a whole host of perceptions, and misconceptions" (Rowlinson, 1988) regarding the design-build approach. In a sense, the differences between traditional and ISD projects might not be so important in a technological sense per se. Rather projects can be differentiated based on the contracting process. Design-bid-build projects allows the separation of design and build responsibilities (Marshall, 1992) while design-build projects confuse design and build responsibilities, which leads to confusion in project execution and accusations of "preferential engineering". While traditional projects have the potential choice between design-bid-build and design-build, ISD projects do not (Section 8.6). It remains to be seen whether the lessons of ISD projects will help to settle the debate in the construction industry.

In summary, the comparative study in this section has highlighted that the nature of software product demands a different project management approach from that for a construction product. The separability of design and build potentially enables the engagement of building contractors to be firmly based on approved designs in construction projects. Software development has no equivalent detailed design to rely on when it comes to contract suppliers or to verify production progress or product quality. These differences make the construction metaphor a misleading one for software development in ISD projects.

9.4 Comparing ISD with Manufacturing Product Development

The contrastive analysis in Section 9.2 suggests that competition in product development helps to increase product quality and control development cost and schedule. However, it has

to be admitted that the mass-market competitive mechanism for PPSW does not suit COTS/CBSW acquisition precisely because COTS/CBSW products provide clients with opportunities to be involved closely for meeting their specialized needs. The question is therefore whether there is a mechanism to enable close client-supplier interaction without the risk of monopolised development. This section examines one such possible mechanism based on the selectionist approach used for new component development in the manufacturing industry.

9.4.1 Design versus Manufacturing

Traditionally, manufacturing products have been regarded as entirely different from software products. Commenting on software's changeable nature, Brooks (1995) states:

> "The software entity is constantly subject to pressures for change. Of course, so are buildings, cars, computers [sic]. But manufactured things are infrequently changed after manufacture; they are superseded by later models, or essential changes are incorporated in later serial-number copies of the same basic design." (p. 184)

It may well be valid for Brooks to attribute some of the software failures to software products' apparent changeability relative to manufactured goods. However, it has to be argued that it is misleading to compare them directly. This is because a software product is the output of a *creative* process that is characterised by uncertainty (Section 8.7). On the contrary, manufacturing is a *copying* process demanding conformity of the output to an approved design. Manufacturing products do not have to change much largely because necessary conceptual changes have taken place in the process of product design. The design process of a manufacturing product, however, is often as creative and uncertain as software design and development, involving marketing, engineering, suppliers and potential customers among others. Therefore software design and development should be compared with the product design process, not its manufacturing process.

There are many types of product design processes in the manufacturing industry.

9.4.2 "Black Box" Component Development in Manufacturing

Typical manufactured products like cars and mobile phones are assembled from many components. Such assembled products may be called System Level Manufactured Products (SLMP), in contrast to the components they are made of. In modern SLMP assembly factories, most, if not all, components required are purchased from external suppliers. How the components are made and which suppliers are contracted to supply them is initially decided in the new product development stage.

In studying the new product development process of twenty automobile makers across Japan, Europe and North America in 1980s, Clark and Fujimoto (1991) describe three different types of client-supplier relationship:

- Supplier proprietary parts: standard catalogue items sold to the car maker (the client) with no modification;
- Black box parts: requires developmental input from both the client and the supplier;
- Detail-controlled parts: all engineering work is done in-house by the client, with suppliers being little more than providers of production capacity.

The first type does not involve the client in the process of design, while the third type does not involve the supplier. The development of the second type, the "black box parts", is of particular interest here due to the contribution of both the client and the supplier to designing the component. Such a process is similar to software design and development in that it requires close client-supplier interaction yet the design outcome may be uncertain. Clark and Fujimoto (1991) describe the black box component development process as follows (the assembler refers to the car maker):

> "Typically an assembler generates cost/performance requirements, exterior shapes, interface details, and other basic design information based on the total vehicle planning and layout [...].
>
> In the Japanese case, requirements information is usually communicated to two or

three potential suppliers that compete for the job. This small group of suppliers usually does the engineering for a particular component on all the auto maker's products. This selection process, called "development competition", typically takes from six to twelve months. In some cases, suppliers may not wait for inquiries from auto makers. Based on their knowledge of emerging technologies and product cycle plans, they may initiate development actions and suggestions. Once chosen, the supplier carries out detailed engineering, including drafting, prototyping, and unit testing. The car maker reviews parts drawings, tests the parts in prototype vehicles, ensures that requirements are met, and approves the design -- thus the Japanese term for this arrangement, 'approved drawings (Shonin-zu) system'." (pp. 140-142)

From the above description it can be seen that a), the client contributes "basic design information"; b), the requirements information is issued to more than one supplier for "development competition"; and c), a winning supplier is *selected* by testing and approving its design. Suppliers assume a considerable level of risk in the "development competition" stage that may last six to twelve months. Clark and Fujimoto further comment on the black box system:

> "Effective management of the black box parts system requires careful balancing. Assemblers have to juggle long-term relationships with suppliers against the need to keep the market competitive by inviting other suppliers to participate in "development competitions". Assemblers also must balance the need to retain key component technologies (e.g., electronics) in order to effectively assess design quality while providing technical support to suppliers and controlling basic engineering for total vehicle integrity [...]." (pp. 142-143)

In the manufacturing industry in Japan where supplier partnership is supposed to be the key to its success (McGloin and Grant, 1997), what characterises the black box component development process is "development competition"! Such a development process is possible due with frequent information exchanges between the client and suppliers based on "partnership" (Dyer and Ouchi, 1993; Karlsson et al., 1998). With careful management, competition between suppliers becomes a mechanism to facilitate client-supplier partnership which cannot be characterised as "adversarial" as has often been perceived (see e.g. Brown, 1997; Lamming and Cox, 1995, quoted in Parker and Hartley, 1997).

9.4.3 Lessons from the SLMP Design Process

There are similarities between the "black-box" component development process and software

development in ISD projects because both require input from the client and innovation from the suppliers. The client works with the suppliers by issuing only "basic design information", which can be regarded as the "revealed requirements" according to the Kano Model (Section 8.3). One important feature of this process as described by Clark and Fujimoto is that there is little interference from the client regarding how the suppliers carry out the design, hence the term "black-box". The client relies on development competition for results first and then adopts the selectionist approach to obtain the best product. However, it does not mean that the relationship between the client and the suppliers is antagonistic. On the contrary, the client works with suppliers closely to realise the design. It might be said that in this case supplier partnership is being built on top of a competitive mechanism. Suppliers compete with each other to work with the client proactively.

Of the three types of client-supplier relations described by Clark and Fujimoto (1991), none fits the competitive-bidding-monopolised-development pattern observed in ISD projects. It is difficult to imagine that "auto makers" would enjoy the same level of proactive contributions to component development if a supplier only starts to commit development resources after having won the competition first. Development competition allows a client to harness suppliers' technical expertise with limited risk. Of course, suppliers are willing to play the development competition game due to the prospective reward of volume orders. However, even after product design approval, suppliers have to be competitive in terms of unit price and product quality to be awarded such volume orders.

The development competition cannot be applied directly to ISD projects, since for CBSW/COTS development, there is no equivalent prospect of volume business as an incentive for suppliers to participate (for PPSW, replicating copies might be a close equivalent). A supplier is unlikely to be willing to participate in a development competition without being compensated for the opportunity cost. This does not mean that the comparison

is necessarily invalid. It does mean that a different incentive mechanism has to be devised if development competition is to be adopted for CBSW/COTS development in ISD projects. Chapter 10 suggests a partial funding approach as part of a new management framework.

9.5 Open Source Software Development

Software development is a complex process that requires the co-operation of highly skilled personnel. A question might be posed regarding the technical feasibility of a competitive mechanism for developing software. Would competition between developers cause confusion? While there is certainly a need for further research into how a competitive mechanism can be harnessed effectively for commercial ISD projects (see Chapter 10 for a suggested framework), development competition has already been at work successfully for developing Open Source Software (OSS). This section examines the OSS development process to see how it helps improving the quality and speed of OSS development.

Open Source Software (OSS) refers to software developed and distributed under one of the Open Source licences that permit free distribution and require availability of the source codes (Linuxcare, 2000). There have been some notably successful OSS products like Linux and Apache web server (Zimmerman, 2001). OSS is cost-effective and has been gradually accepted by large organisations both in public and private sectors. According to Sol (1998), Microsoft had assessed OSS' quality in its internal memorandum:

> "Recent case studies (the Internet) provide very dramatic evidence [...] that commercial quality can be achieved/exceeded by OSS projects."

OSS phenomenon has been subject to many studies from different perspectives. For example, Lerner and Tirole (2000) explore the economic rationale for OSS' very existence. Kenwood (2001) evaluates OSS' potential to the development of military systems for the US army. From a development process viewpoint, OSS is unique in many ways. First of all, OSS is developed with software programmers scattered across the globe. Secondly, there is no

formal, disciplined requirements management process (Scacchi, 2001). Thirdly, a process of competitive development is used with stringent output evaluation and selection. There are many other aspects worth commenting on but it is the competitive development that is relevant here.

9.5.1 OSS Development Process

Kenwood (2001) notes that "[o]pen source projects utilize multiple small teams of individuals that work independently to solve specific problems [...]. Parallel debugging and development efforts enable the coordinating developer to choose the best potential implementation from the many choices offered" (p. 7). What lies behind the OSS process is the principle "Let the best code win", apparently adopted by Linus Torvalds, the founder of Linux Project, one of the archetypal OSS projects. Pavlicek (2000) elaborate it in this way:

> "It does not matter whether the best solution comes from someone with three PhDs and thirty years of experience or from a bright 18-year-old who is just beginning college. The project will benefit the most by using the best solution. The pedigree of the author makes little difference. The issue is the solution." (p. 15)

In addition to multiple solutions available for evaluation, the evaluation itself is given a high level of resources. The evaluators are highly qualified core-developers supplemented by anyone with an interest in the project. Mockus et al. (2000) describes the process:

> "All of the core developers are responsible for reviewing... to ensure that the changes are appropriate. In addition, since anyone can subscribe to the mailing list, the changes are reviewed by many people outside the core development community, which often results in useful feedback before the software is formally released as a package."

Through a voting system, multiple evaluations contribute to updating a "STATUS file" for a relevant "action item" (Apache, undated-a). The evaluation process is a selection process. It produces candidates for inclusion in the next release. A Release Manager (RM) then decides what to include based on the information available in the STATUS files. RM seems to have the total discretion in deciding what to include and what to exclude in making up a release. In the Apache Project guideline (Apache, undated), it is stated that:

> "Regarding what makes it into a release, the RM is the unquestioned authority. No one can contest what makes it into the release. The community will judge the release's quality after it has been issued, but the community cannot force the RM to include a feature that they feel uncomfortable adding. Remember that this document is only a guideline to the community and future RMs - each RM may run a release in a different way. If you don't like what an RM is doing, start preparing for your own competing release."

The firm authority given to the Release Manager should prevent endless debates about the pros and cons of taking one action over another. Such authority allows the Release Manager to make quick decisions in addition to maintaining a consistent standard. Kuwabara (2000) gives developers' accounts of how Linus Torvalds, the keeper of Linux kernel, maintains coding and stylistic standards:

> "With a growing developer base, he has naturally become more selective in regards to accepting new code for the kernel [...]. Torvalds enforces not only quality standards, but stylistic standards also. Several developers recounted their personal experiences, in which a patch they submitted was rejected, simply because it was too complex."

With such stringent quality control, it is not surprising that OSS products like Linux and Apache web server have been highly successful (Mockus et el., 2000). Enforcing quality standards and stylistic standards is a luxury a commercial project does not have due to the pressure of project deadlines and the lack of realistic options (Section 7.5; Section 8.5).

One effect of OSS development process is its speed of response to defects. In the process of developing software products, defects are often considered inevitable (Davis, undated; Ganssle, 2002). The response time is critical in reducing users' waiting time. OSS distributes fixes and patches quickly, "potentially an order of magnitude faster than those of commercial software (Kenwood, 2001, p. xiii). Such a fast turnaround time has a lot to do with the shear number of developers participating in OSS projects. It may also be related to the inherent competitive development mechanism that *selects* solutions.

OSS projects are not guaranteed successes (Zimmerman, 2001). Failures may be attributed to a number of reasons. One possibility is the lack of financial incentives. The other is the lack

of technical capacity on the Internet to accommodate too many OSS projects. Or an OSS project can pursue a wrong strategy (Zimmerman, 2001). The development process itself does not appear to have been questioned. A thorough study of the causes of failed OSS projects requires their background knowledge and comparative analysis with their suitable contrasts. This is outside the scope of the thesis and may be appropriate for a separate study.

9.5.2 Capitalising on Software Virtuality

The fact that software is virtual rather than physical has caused many problems in software development (Section 9.3). Not only has it hindered communication between client and suppliers and among users, software designers and developers (Brooks, 1995), it may have also helped suppressing the investigation of failed software projects. Had the billions of pounds or dollars been wasted on physical product development, there would have been much more visible "debris" requiring investigations.

OSS process, however, has turned software virtuality to its advantage. The fact that more than one copy of a software component is submitted with only one being selected does not appear to have caused any concern of excessive software programs being produced. Developers keep developing OSS components partly as a learning exercise. So far the value of learning exceeds that of the effort of development, the effort of development is justified. The cost structure of software means that rejected software can be thrown away without any debris, or in economic terms, negative externality (Livesey, 1993). This is different for a physical product. Physical product development usually involves purchasing materials. Not only the materials cost money directly to firms undertaking development, the rejected materials may incur cost to be recycled or cause negative externality to the environment.

In summary, OSS uses a unique process to develop complex software products. Software artefacts are developed competitively by developers in parallel. Quality control is achieved by

selection or relative product evaluation, not by enforcing a list of acceptance criteria. The OSS process has also capitalised on software virtuality. Though OSS process will not be directly applicable to commercial ISD projects due to its lack of any commercial dimension for rewarding software programmers, it demonstrates the technical feasibility of parallel development, possible improvement of software quality and reduction of development lead-time.

9.6 Summary

This chapter first examines the validity of the proposed solution with a competitive mechanism. Section 9.2 contrasts CBSW/COTS development in ISD projects with PPSW development. The analysis suggests that it is due to a competitive market mechanism that PPSW suppliers are highly incentivised to produce quality products. Though PPSW applications may also be delivered late and with poor quality initially, clients are protected from suppliers' failures by having a chance to evaluate first before making purchasing commitments. A counter argument against this explanation might be envisaged considering the monopolised product development in construction projects. Section 9.3 examines the differences between software development and construction. The physical nature of construction products and the maturity of the construction industry have made it possible to separate design and development. A detailed architectural design serves as the basis of builder contracting, project control and monitoring, and the final product acceptance. The inseparability of software design and development, on the other hand, means that there is no equivalent detailed design before supplier contracting. The conclusion is that, as successful as construction projects often are, they do not provide an appropriate metaphor for ISD projects.

While the competitive product development mechanism might be desirable, is it feasible with a client being closely involved in controlling requirements? And is it technically feasible for software development? Two analogical studies are carried out in an attempt to answer these

questions. First of all, the "black-box" component development process in the manufacturing industry is examined to see how competition allows a client to control requirements and manage development uncertainty at the same time. The process is characterised by "development competition", with the client only providing "basic design information". The client relies on selection to obtain the best design. The technical feasibility of development competition in software development is demonstrated by the OSS development process. Through product selection, OSS process produces high quality products with fast turnaround time for enhancement and defect fixes.

Chapter 10: Performance-Based Systems Development Framework

> Any prince who has come to depend entirely on promises and has taken no other precautions ensures his own ruin...
> (Niccolo Machiavelli, *The Prince*, Chapter XVII)

10.1 Introduction

The unified explanation given in Chapter 8 for ISD project challenges implies a competitive development mechanism. Chapter 9 further suggests that the given explanation is a warranted inference that might be useful for guiding future actions. However, the competitive product development processes for PPSW, OSS and manufacturing components discussed in Chapter 9 are not directly applicable to ISD projects. The development process for PPSW products does not give clients the chance to control requirements closely. The OSS process does not have a commercial dimension to engage suppliers. The process for manufacturing product development relies on volume business to incentivise suppliers to participate. To implement the competitive process for software development in ISD projects, it is necessary to devise a new management framework. In this Chapter, such a framework is proposed based on the unified explanation to address the weaknesses of TSDF. The framework is called Performance-Based Systems Development Framework (PBSDF). PBSDF provides one possible way to test the proposed explanation empirically. Section 10.2 outlines some of the essential understandings required to implement PBSDF. Section 10.3 gives an outline of PBSDF itself, consisting essentially of three parts: outcome-based contracts, development competition and relative product evaluation. A key concept in PBSDF is the Non-Acceptance Price Factor, which is discussed in Sections 10.4 to 10.6. The implementation of PBSDF with two suppliers is discussed in Sections 10.7 to 10.9. Alternative implementations of PBSDF with a single supplier and multiple suppliers are discussed in Section 10.10. A number of other issues are discussed in Section 10.11 followed by a summary in Section 10.12.

10.2 Pre-Contract Understandings

An effective management framework should prepare a client for failures yet increase the chance for success. PBSDF aims to meet the following criteria:

- To enable software product development to be driven by client's requirements, not by suppliers' delivery (Section 7.4.1 and Section 8.3);
- To achieve the above cost-effectively;
- To achieve the above time-efficiently;
- To manage development uncertainty (Section 8.7) with an exit strategy for both the client and the supplier if developed software products are not fit for use.

PBSDF has been proposed to satisfy the participation constraint and the incentive compatibility constraint in terms of supplier contracting (Section 8.5). The essence of a performance-based management framework is in linking pre-contract commitments and post-contract delivery. As part of PBSDF, it is essential to have the following understandings between the client and the supplier before a software supplier is selected and contracted. The understandings cover a number of areas including the concept of a software system, requirements management, possible variations of price and delivery time, possible outcomes of product development etc. Each of these understandings is discussed below.

10.2.1 Pre-Contract Understanding: The Concept of An Information System

An information system is made up of a system platform (including hardware, system software and necessary networking facilities), one or more software application (including appropriate documentation) and data (see the definition of an information system in Section 3.2). While the required interoperability of an application and the system platform causes little dispute, a particular emphasis is to be drawn toward the interoperability of a software application and the operational data. One essential understanding between the client and the supplier must be that a software application, if it is to be considered acceptable, must be able to work with the variety and volume of the client's data. This is particularly important when there are legacy

data to be incorporated into a new system. While the client must provide the supplier with its data requirements (either in the form of legacy data or a full data definition), the supplier must deliver a software product fully compatible with client's data requirements.

10.2.2 Pre-Contract Understanding: Incomplete and Changing Requirements

According the Kano Model (Section 8.3), the client can only provide "revealed requirements" while leaving the supplier to fill in with "expected requirements" and "exciting requirements". At the start of an ISD project, the client does not necessarily have a full set of "revealed requirements". In the process of development, due to improved understanding of either the requirements, or design and development, and sometimes, due to external changes (business requirements, regulatory environment etc), the "revealed requirements" are likely to go through a series of changes. Assume the initial requirements are referred to as R_0, and there are m subsequent changes ΔR_1, ΔR_2, ΔR_3, ... ΔR_m. With ΔR_i representing the ith change, the overall requirement R is:

$$R = R_0 + \sum_{i=1}^{m} \Delta R_i$$

It should be noted that: a), client requirements are about what is required, not how they are to be met; b), it is conceivable that later requirements changes may displace or cancel out earlier ones; c), requirements changes may be proposed by the supplier as well as the client (particularly if scope reduction is requested reflecting design and development reality); d), each change, whether it has any material effect on cost or delivery time, must be documented and communicated in writing to relevant suppliers; and e), the supplier's task is to provide a software product to meet or exceed R, not just R_0.

10.2.3 Pre-Contract Understanding: Price Changes

At any point of software development, there is one aggregate total price P between the client and the supplier. P may change as software development unfolds. P may increase or decrease

depending on requirements changes, contract re-negotiation or project delays etc. P can be expressed as a sum of the initial price and subsequent price changes. Assume the initial price is P_0, and there are n subsequent changes to the price ΔP_1, ΔP_2, ΔP_3, ... ΔP_n. With ΔP_j representing the jth change, P is:

$$P = P_0 + \sum_{j=1}^{n} \Delta P_j$$

Like requirements changes, the initial price and the associated assumptions and the subsequent price changes must be fully recorded.

10.2.4 Pre-Contract Understanding: Delivery Time Adjustments

Upon starting an ISD project, a client may well have in mind a target project deadline. Accordingly, there is a target delivery date for any software product from a supplier, taking into consideration of the time required for product evaluation, integration and implementation. If T is designated as the total length of delivery time required for a supplier to deliver a working software product, T will change as project progresses both due to requirements changes and development uncertainty. Let T_0 be the initial delivery time promised by a supplier, and assume there are w changes to the delivery time ΔT_1, ΔT_2, ΔT_3, ... ΔT_w. With ΔT_k designated as the kth change, the delivery time as measured from the development start date will be T:

$$T = T_0 + \sum_{k=1}^{w} \Delta T_k$$

The software development start date is relatively straightforward to determine. Having selected a supplier, and perhaps after a period of post-selection negotiation, the client instructs the supplier to start design and development. This signifies the end of a supplier's sales effort and the start of the delivery effort. Software product delivery date (PDD) is the supplier's

promised date to submit the software product in its final form for evaluation. Prior to PDD, there may well be initial versions of software delivered for demonstration, testing or training purposes. The software product delivered on PDD, however, should be fully quality-assured by the supplier with a full set of relevant documentation. The client makes an acceptance decision based on evaluation of the delivered product on PDD.

10.3 An Outline of PBSDF

PBSDF consists of three parts. Part I is a contract between the client and the supplier, not based on complete requirements but based on the possible outcomes of product development and evaluation. Part II is a development competition phase that is key to manage development uncertainty and any possible change in the post-contractual phase. Part III is a stage for product evaluation.

10.3.1 PBSDF - Part I: A More Complete Contract

When a client and a supplier enter into a software development contract in the beginning of an ISD project, the outcome may be uncertain. According to the discussion in Section 8.5, a more complete contract can be achieved between a client and a supplier if the eventualities of the development is understood and planned for with appropriate payoff contingencies. For the purpose of describing PBSDF here, the following five scenarios are envisaged:

- Scenario 1 (S1): The supplier has developed a working software product by a pre-agreed PDD. The product meets or exceeds the client's needs and/or expectation, and is therefore accepted by the client;
- Scenario 2 (S2): The supplier has developed a working software product by PDD but the product is not accepted by the client for some reason (e.g. not meeting the client's expectation). Client wishes to stop further engagement of the supplier;
- Scenario 3 (S3): The supplier has not developed a working software product by PDD. Client wishes to continue development and agree a new delivery date (plus possibly

extra cost);

- Scenario 4 (S4): The supplier has not developed a working software product by PDD. The client wishes to stop the development further at that point;
- Scenario 5 (S5): The client cancels the development before PDD.

The five scenarios are illustrated in Figure 10.1.

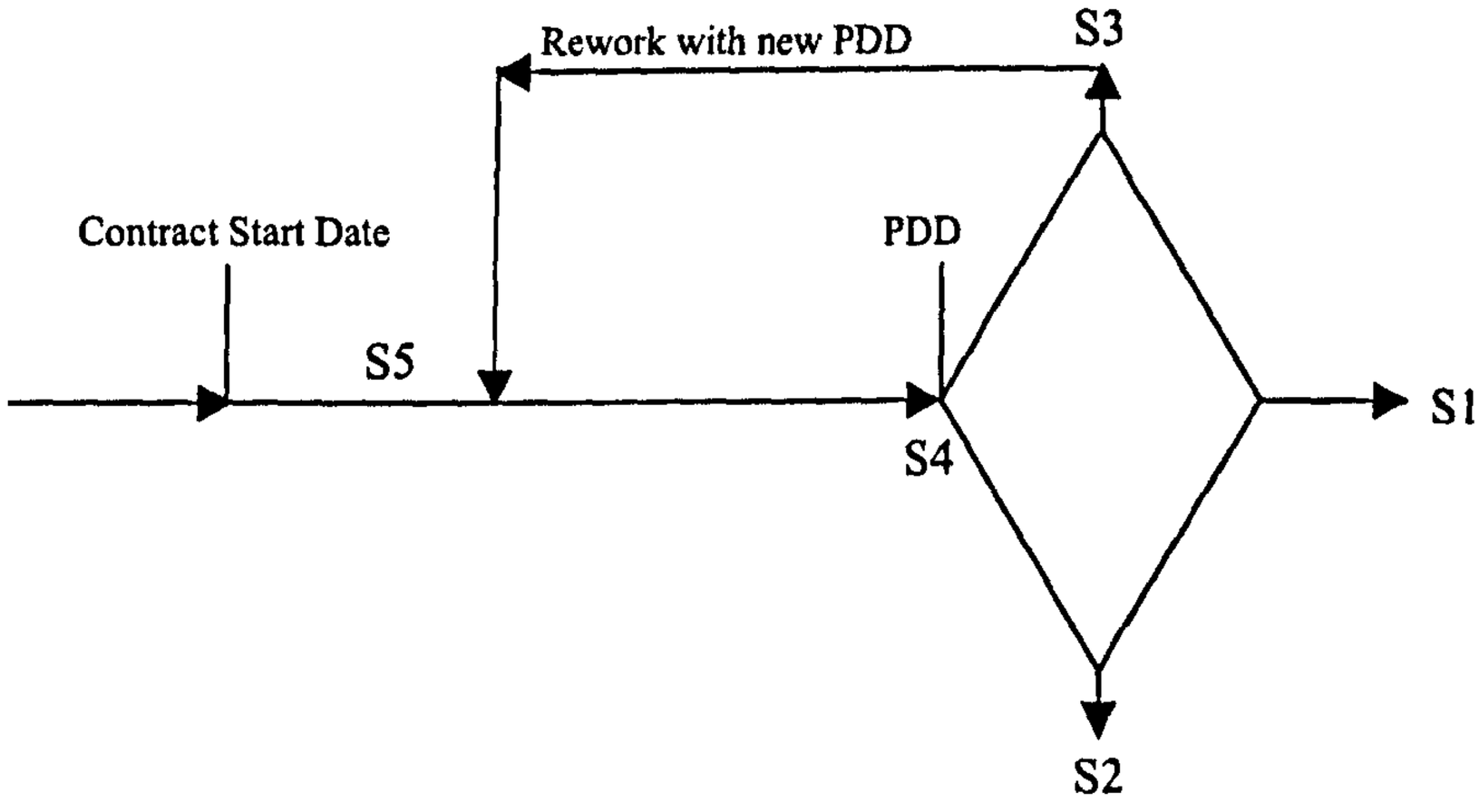


Figure 10.1: Five possible outcome scenarios of a software development contract

Other possibilities exist for different projects in different circumstances. For example, a supplier may request to cancel a contract before PDD. All potential scenarios should be planned for as part of the contracting process. The following discussions will focus on the above five generic ones. In the above description, a "working software product" is an important concept that can be defined with certain flexibility to suit different circumstances. Generally speaking, "a working product" should meet the following criteria: 1), the software product meets the revealed requirements in terms of functions and is delivered with appropriate documentation; 2), it can be installed successfully onto the target system platform with minimum help from the supplier; and 3), it should compile on target system platform and operate with the test data from the supplier successfully.

In the above five scenarios, S1 is the desired outcome at the point of contracting. With S1, the client pays the supplier the full price *P*. S3 is not an end status. It is conceivable that S3 may

develop into any of the other scenarios. In TSDF, software projects are assumed to proceed smoothly to the outcome of S1 while S2, S4, and S5 are not planned for. In PBSDF, the five scenarios should be planned for as a minimum. To cater for S2, S4 and S5, three price factors should be agreed between the client and the supplier in addition to the full purchase price. These are Non-Acceptance Price Factor, Non-Delivery Price Factor and Development Cancellation Price Factor.

Non-Acceptance Price Factor

The Non-Acceptance Price Factor (NAPF), designated as λ, is defined as the percentage of the total price the client is willing to pay and the supplier is willing to accept if the software product delivered is rejected by the client. In other words, when S2 occurs the client pays the supplier a non-acceptance price P_{na}:

$$P_{na} = \lambda * P$$

Where $0 \leq \lambda \leq 1$. The agreement of λ is preferable to a fixed Non-Acceptance Price since the total price for a software product is likely to change throughout the development process. The value of λ will stay the same, however, regardless the changes in requirements, the overall price and delivery time. The agreement of λ between the client and the supplier before the contract is instrumental to PBSDF. Section 10.4 provides a theoretical discussion on setting λ and Section 10.5 provides empirical estimates accordingly. In Section 10.6, some additional considerations are given for setting λ within the context of ISD projects.

A question may well be raised regarding possible reasons for a client to reject a supplier's delivered product. Should some differentiation be made regarding if the rejection is due to the failure to meet the client's requirements or the failure to meet the client's expectation? In PBSDF, however, this distinction is not pursued due to the understanding that a client's initial requirements are not complete in the first place. More importantly, the client is given a binary

decision that a software product is either used or not used. A software product may be used if it is judged to bring enough value to the client to justify its adoption even if it does not meet the initial requirements. Likewise, a software product may be rejected even if it exceeds the initial requirements and expectation due to the availability of a better product. The key issue is not if a product meets the requirements specification. It is whether the client thinks that using the product provides more value than not using it, comparing with alternatives available. Here the fundamental assumption that the contracting parties are economically rational is important (see Sections 8.2 and 8.9.2). The client must be trusted to make such a decision rationally and the supplier must make a judgement regarding the client's rationality before contracting with the client.

For the client, the non-acceptance payment of λP can be considered as an insurance premium against obtaining a system that might bring less value than otherwise possible. This premium allows the company to cancel the contract with the supplier. On the other hand, a supplier expects to cover part of its actual cost and some opportunity cost incurred. λP represents the minimum the supplier is prepared to accept in exchange of its commitment of resources for undertaking a risky venture. The value of λ can be derived from the marginal cost pricing principle in microeconomics. This is discussed in Sections 10.4, 10.5 and 10.6.

To base the product acceptance decision on the value contribution has the benefit of linking software development to value engineering for the client organisation. According to Morris (2000), there is a research gap in the value engineering and value management in the project management discipline. The PBSDF approach might bring a useful perspective in this regard. The subject might be pursued further as a separate research area (Section 11.6.6).

Non-Delivery Price Factor

Similar to the Non-Acceptance Price Factor, a Non-Delivery Price Factor, designated as μ,

may be agreed to cater for S4. In this scenario, a supplier fails to deliver a working product by PDD, the client may cancel the development contract and pay the supplier a non-delivery price P_{nd}:

$$P_{nd} = \mu * P$$

Where $0 \leq \mu \leq \lambda$. The fact that $\mu \geq 0$ means that no consequential damage is claimed for non-delivery of software product. It is conceivable that $\mu = 0$ if there is enough competition in the product supply market. The fact that $\mu \leq \lambda$ should encourage the supplier to complete development rather than to abandon it half way.

Development Cancellation Price Factor

The scenarios of either non-delivery or non-acceptance only take place on or after PDD. However, there is a possibility that the client may wish to terminate software development before PDD. To cater for such an eventuality, the client may agree a Development Cancellation Price Factor, designated as ν. If the client cancels development before PDD, the supplier is paid a development cancellation price P_{dc}:

$$P_{dc} = \nu * P$$

Where $0 \leq \nu \leq \lambda$. The value of ν may be linked to the length of time already spent for development or linked to the actual cost already incurred by the supplier.

10.3.2 PBSDF – Part II: Development Competition

PBSDF adopts development competition to manage development uncertainty and the associated risks. Within PBSDF, development competition can be defined as engaging two or more suppliers in designing and development products to the same revealed requirements from the client. The technique of development competition is demonstrated in Section 9.4.2 for developing "black box" components in the manufacturing industry. The concept is clarified further here. In a broad sense, development competition is similar to the concept of

"managed competition". There appear to be at least two different uses of the term "managed competition". One refers to a mechanism to introduce competition into the provision of public services, especially public health services (see e.g. Shortliffe and Perreault, 2000). The other refers to a form of competition that is introduced to an otherwise regulated monopoly (Wolinsky, 1997; Suzuki, 2002). Development competition is effectively managed competition used for product development. Suzuki (2002) studies managed competition in various forms in the manufacturing industries, including development competition to produce new automobile components as discussed in Section 9.4.2. Development competition has also been used to manage mission-critical product development in projects. For example, Zimmerman (2000) reports the development of "the next-generation combat weapon for the U.S. forces" by two suppliers' direct competition with a successful result. Zimmerman concludes that premature selection of a winning supplier in product development amounts to "gambling". In addition, competition "brings out the best from the contractors, reduces program risk and maximises technology options for the customer" (Zimmerman, 2000, p. 67). These lessons are applied here in devising PBSDF for software development.

One important feature of development competition in manufacturing industries is that rewards for the winning suppliers are (mainly) in the form of volume orders at the production stage, not directly from winning the competition (Section 9.4). There is no equivalent volume business in ISD projects if only one instance of software installation is required. To encourage suppliers to participate, it is necessary that the client partially funds the development activities. This happened in Zimmerman (2000) in that the client contributed $10 million to each supplier up-front for product development, though the rationale of providing this particular level of funding was not discussed. The level of partial funding may be reduced if volume installations are planned for. In that case, suppliers' rewards, at in part, can be tied to the number of installations. Such a mechanism is emphasised in implementing PBSDF with multiple suppliers (Section 10.10.2).

Development competition reduces risks considerably for the client. Literature on competition suggests, however, that a supplier can derive considerable benefits from competition too. Porter (1985) describes in detail the strategic benefits of having competitors. For example, he suggests that competitors enhance a supplier's ability to differentiate itself:

> "Without a competitor, buyers may have more difficulty perceiving the value created by a firm, and may, therefore, be more price- or service-sensitive. As a result, buyers may bargain harder on price, service or product quality. A competitor's product becomes a benchmark for measuring relative performance, however, which allows a firm to demonstrate its superiority more persuasively or lower the cost of differentiation." (p. 203)

Porter goes on to point out many other benefits a viable competitor brings, like increasing a supplier's internal motivation, increasing demand, reducing buyers' risk and promoting the image of the industry etc. Many of these are applicable to software development.

10.3.3 PBSDF – Part III: Product Evaluation

Section 8.8 discusses the problems with product evaluation in TSDF, especially during the product acceptance stage using the absolute product evaluation approach. The absolute product evaluation measures a product against an idealised model (Morisio et al., 2002) or standards (Blin and Tsoukias, 2001). In TSDF, the absolute model is often taken to be the initial client requirements rather than a complete detailed design. This makes the product evaluation in TSDF costly and problematic (Section 8.8).

PBSDF provides the possibility for the client to use the relative product evaluation approach at product acceptance stage. The term "relative" has two meanings. First of all, in PBSDF, the client does not make the product acceptance decision based on whether a product meets the initial requirements that would be incomplete anyway. Instead, the evaluation is *relative* to the value of the product perceived by the client organisation. This is possible in PBSDF due to the fact that the contract caters for a scenario that the developed product may be rejected

for a non-acceptance payment. If there is only one product to be developed, and the product is rejected upon evaluation, it can be interpreted that the product would bring less value to the client than if adopted, *ceteris paribus*. Though such a process sounds subjective on the client's side, the subjectivity is limited due to ongoing requirements adjustment in PBSDF. Product evaluation relative to the client's value judgement provides a crucial link between software development and the client's legitimate value expectation.

The second reason for the product evaluation in PBSDF to be called "relative" is that with development competition of two or more suppliers, each supplier's product can be evaluated *relative* to other comparable product(s). This is particularly helpful on dimensions of functionality and quality where absolute measurements are difficult like design architecture, usability and security. The technique can be equally effective when applied to measurable dimensions like system performance. The relative approach does not exclude the measurement of quality dimensions or the setting of some initial quantified performance expectations. The key point is that it does not purport to measure an end product against a complete and idealised product model.

10.4 The Theoretical Basis of λ

The Non-Acceptance Price Factor λ is set to incentivise suppliers to participate. If a supplier makes an honest bid, and makes the best effort to produce the software that gets rejected in the end, what would be the minimum amount of payment a supplier is willing to accept for participation? The answer might be found in the marginal cost pricing principle in microeconomics. According to this principle, in a competitive market, a supplier expands its production until its marginal revenue is equal to its marginal cost (Lumsden, 1995; Parkin et al., 1997). This principle applies either for short term or long term pricing. In the short term, the marginal cost can be approximated with variable cost. In other words, so far as the supplier's variable cost is covered, a supplier may accept orders, even at a price below the

normal long-term selling price (Lothian and Small, 1991). The same idea is expressed as the "loss avoidance" principle as observed by Keser and Willinger (2000) in studying payment schemes. Accordingly, in the short term at least, a software supplier is likely to agree to participate in a software product development if its variable cost is paid for by the client for the eventuality of its product not being accepted, with an expectation to be paid a full price if its product is accepted. Assume that the opportunity cost is zero (the opportunity cost is addressed in Section 10.6 as part of practical consideration when setting λ), a supplier's total quoted price P may be broken down into various costs and profit:

$$P = \text{variable cost} + \text{fixed cost} + \text{supplier profit}$$

The Non-Acceptance Price Factor λ can thus be defined as:

$$\lambda = (\text{variable cost})/P$$

In traditional manufacturing industries, the variable cost is made up of raw materials, direct labour and the variable portion of overhead cost (Lothian and Small, 1991). To develop a software product, the cost for raw materials is negligible. The variable cost for the software product can be approximated by the direct labour cost (DLC) for the personnel directly involved in the development. DLC for software development includes basic salaries plus any pension contribution and tax liability for those directly involved in development including analysts, developers, testers and project managers, but not for line managers and other support staff. The invariable cost would cover all overhead costs including salaries and benefits for line managers, support staff and expenses relating to the office rental, heating etc. Depreciation is normally treated as a fixed cost since it has more to do with the passage of time rather than the actual usage (Lothian and Small, 1991). Therefore, the Non-Acceptance Price Factor λ for a software development contract can be estimated as:

$$\lambda \approx \text{DLC}/P$$

The next section makes an attempt to estimate λ based on the above definition.

10.5 Empirical Estimates of λ

In this section, discussions focus on using two sources of data for estimating λ. The first is published annual accounts of software companies. The second is data from SLC in the participant case studies (Chapters 4, 5 and 6). Further practical considerations in setting λ are discussed in Section 10.6.

10.5.1 Estimating λ with Data from Annual Accounts

All publicly quoted software development companies disclose accounting information in their annual reports. The information is usually available at a high level. Even so, it is a starting point to obtain a rough estimate of λ. Consider the three software companies listed in Table 10.1. These companies are selected because their published accounts provide details about "service revenues" versus the "cost of services" while many companies do not separate information about services and product (e.g. PPSW products) sales. Since product sales have a different cost structure compared with services, such data have to be excluded. The "cost of services" can be interpreted as the variable cost for the corresponding "service revenues". Unfortunately, the service revenues and costs listed in Table 10.1 may include services like consulting, training and support in addition to any custom software development activities. However, it is assumed that service charge rates on average are comparable to those for software product development, if not higher. The data can provide an indication of the high end of the variable cost as a percentage of the corresponding revenue.

Table 10.1: Service costs as percentages of the corresponding service revenues

Companies		1999	2000	2001
Business Objects (www.businessobjects.com)	Revenues from Services ($000)	87,896	128,089	166,200
	Cost of Services ($000)	35,467	53,101	63,497
	Cost as percentage of revenues	40.4%	41.5%	38.2%
Quest (www.quest.com)	Revenues from Services ($000)	16,599	38,820	71,216
	Cost of Services ($000)	4,195	10,695	17,369
	Cost as percentage of revenues	25.3%	27.6%	24.4%
Mercury Interactive (www.mercuryinteractive.com)	Revenues from Services ($000)	56,800	90,900	124,400
	Cost of Services ($000)	16,957	24,679	29,231
	Cost as percentage of revenues	29.9%	27.1%	23.5%

Table 10.1 shows that the variable cost percentages of their corresponding service revenues range from 23.5% to 41.5%. This range gives a rough indication of the possible λ for these companies. Company annual reports also reveal some information about staff costs. However, costs for direct labour, support staff and managerial staff are usually not separated. To get a more accurate estimate of λ, the next Section takes a closer look at the data from SLC.

10.5.2 Estimating λ by Labour Cost Data from SLC

Being a company offering system integration services to ISD projects, SLC's staff costs and their charge rates to a client are comparable to those of software suppliers. For SLC, the unit cost of using internal resources or internal direct labour (IDL) includes staff salaries, benefits and tax liabilities. The corresponding unit revenue is the IDL's project charge rate (PCR). PCR is what a software supplier charges the client, typically using man-day as a unit. It is usually decided according to the grade of the staff (junior programmer, programmer or senior programmer etc.) and the going market rates for each grade. PCR is used to plan the project budget first and to consume the budget during the project according to man-days used. The man-days charged to the client multiplied by the corresponding daily PCRs give the supplier the total project revenue. To illustrate the proportion of the cost and the revenue for various types of resources, the figures from Case 1 are shown in Table 10.2.

Table 10.2: SLC's direct labour costs as percentage of PCR (with 23% off the market rates)

	Average Annual Salary Cost (£, including social security and pension costs)	Daily Cost (£, assume 220 working days a year)	Project Charge Rate (PCR, £)	Planned Days as Percentage of Total Project Man-Days (%)	Direct Cost Percentage of revenue (%)
Senior Project Manager	45,069	205	818	15%	25%
Project Manager	37,258	169	681	9%	25%
Senior Technical Specialist	36,821	167	618	10%	27%
Senior Analyst	38,400	175	603	23%	29%
Senior Programmer	32,048	146	482	34%	30%
Programmer	22,558	103	376	9%	27%

It can be seen from the above table that the internal direct labour cost is around 25% to 30% of the project charge rate. The weighted average is 28%. The PCR quoted in the above table were used in the Years 2000 and 2001. The rates were decided upon first by a benchmark with the market average, and then take 23% reduction due to a long-term agreement between SLC and UKOne. The market average project charge rates would have been higher and thus the direct labour percentage of the total charge rate in the market place would have been lower. If the PCR is adjusted back to the market average rate, the direct labour cost percentages are between 19% to 23% (Table 10.3). The weighted average percentage is 21%.

Table 10.3: Direct labour costs as percentages of the average market full charge rates

	Daily cost (£, assume 220 working days per year)	Average Market Project Charge Rate (PCR, £)	Project Days Percentage in a Plan (Case 1-LoEpriceModel)	Direct Cost Percentage (%)
Senior Project Manager	205	1062	15%	19%
Project Manager	169	884	9%	19%
Senior Technical Specialist	167	803	10%	21%
Senior Analyst	175	783	23%	22%
Snr Programmer	146	626	34%	23%
Programmer	103	488	9%	21%

In addition to IDL resources, a supplier may use bought-in contractor labours (BCL) to gain specific skills. BCL may be from self-employed companies, which have been common in the last few years. Or they may come from the so-called "body shops" that lend their staff for a period of time for a fee. The cost for each BCL is typically a daily contract rate with no other liability. On the other hand, the supplier levies the client a project charge rate (PCR), just like the internal resources. PCR is normally higher than the rate the supplier pays for BCL, thus yielding a profit for the supplier. In Case 1, no individual bought-in contract labour was used as defined above. In Case 2, two contractors were used. The contractors' costs to SLC were £300 and £346 per day respectively. The project charge rate was the same for both at £482. This gives the direct cost percentages of 62% and 72%. Using contractor staff is clearly a more expensive option in this case, but still yields gross profit for SLC.

Only the supplier knows the exact direct labour cost percentage of its corresponding revenue. In entering a PBSDF agreement, the client and the supplier share the understanding that the supplier is paid the full price P if its application is accepted, and is paid λP if its application is not accepted. The competitive pressure will induce a supplier to reveal its cost information by agreeing to a level of λ appropriate to its own cost structure. Although the concept of λ is derived from the proportion of variable cost, it may incorporate other practical considerations in practice, some of which are described in the following section.

10.6 Practical Considerations for Setting λ

At the contract bidding stage in PBSDF, a supplier is asked to submit λ together with a full price. Though the full price may change due to many reasons (Section 10.2.3), λ stays the same throughout the project. A supplier has to balance the need to submit a low λ to be competitive and the need to have a high enough λ to cover the risk of its final product not being accepted. The value of λ may vary from 0% to 100% depending on the supplier's other practical considerations. If a supplier sets λ at 0%, the message is that it is confident of its product being developed successfully and accepted. It is equivalent to the conventional "money back" guarantee. If the supplier sets λ to be 100%, the supplier wants to be paid the same whether its product is accepted or not. This is similar to the "cost plus" type of contract when all risks are borne by the client. This section explores a number of practical considerations a supplier might have in setting λ.

10.6.1 Varying λ According to the Level of Competition

If the software development market is fiercely competitive, the competitive pressure may force a supplier to lower the proposed λ value. In the extreme case, suppliers may propose $\lambda =$ 0. On the other hand, if the relevant skills are in short supply and there is a significant lack of

the relevant development capacity, a supplier may demand a higher λ to cover not only the variable cost but also the resource opportunity cost.

10.6.2 Varying λ According to Project Length

It has been assumed that a supplier is willing to embark on a project with a possible prospect of being paid only $P_{na} = \lambda P$ if the development can be accomplished "in the short term" (Section 10.4). In the long run, a supplier must make sure that the fixed costs are recovered through products and services in accordance with the "loss avoidance" principle (Keser and Willinger, 2000). An acceptable PBSDF contract may still be devised for a project lasting longer that "the short term" with λ increased, making larger projects more expensive for the client. Here the concepts of "short term" and "long run" are relative to each other. Traditionally, "short-term" is taken to mean one year in accounting terms. The higher λ value for longer projects would effectively signal to clients to take up smaller projects or break up large projects into smaller chunks.

10.6.3 Varying λ According to Project Risk

A client may undertake ISD projects that rely on cutting-edge technology. That carries higher risks for suppliers. In such a case, a supplier may propose a higher λ. This has the benefit of providing more realistic cost feedback of the risks to the client before undertaking highly risky projects. Such risk feedback mechanism is valuable to the client in evaluating suitable projects to be carried out.

10.6.4 Varying λ According to Supplier's Product Maturity

If a supplier offers a readily available software product that requires little additional development work, λ might be set close to zero. This is simply to capitalise on the fact that the marginal cost of reproducing a software product is low, though implementation cost to

integrate a software product into a successful system should not be overlooked. On the other hand, for customised development, a supplier might set λ to be closer to its true variable cost or even higher depending on other considerations. As a result, a client is more likely to opt for solutions requiring less development.

The flexibility in setting λ value affords PBSDF great flexibility to take account of market condition and the level of project risks. This is in sharp contrast with TSDF that encourages suppliers to bid low initially. Low initial bidding prices entice clients to undertake projects that are otherwise uneconomical. This phenomenon is often described as "not having a valid business case" by some critics of failed ISD projects (Smith, 2001). Similarly, clients might undertake projects that are speculative and not mission-critical, and therefore are more likely to be cancelled. A supplier bidding to participate in such projects may factor in the risk by increasing λ. As a result, some of the non-essential systems may not be funded in the first place. This market feedback mechanism might be considered as a useful by-product of PBSDF. The role PBSDF may play in project feasibility assessment might be modelled mathematically in separate studies (Section 11.6.3).

10.7 Implementing PBSDF with Two Suppliers

This Section provides an illustration of implementing PBSDF using two suppliers engaged in development competition, referred to as the Two Supplier Strategy (PBSDF-TSS, or TSS for short). PBSDF may be implemented with a single supplier without development competition, or implemented with three or more suppliers. These will be discussed in Section 10.10.

In PBSDF-TSS, each supplier participates in contract bidding by submitting a full price and a Non-Acceptance Price Factor λ based on the client's Request for Proposal (the other price factors may be included in the bidding or may be negotiated after the initial selection). Two suppliers are selected to participate in a development competition. Each supplier receives the

same requirements specification from the client and develops a software product independently. Any requirement change agreed between the client and one supplier must be published to the other. Having developed the products, the suppliers submit them for evaluation. The supplier with the winning product is paid its full-asking price. The supplier with the rejected product is paid a λ percentage of its full-asking price. What follows is a description of the possible PBSDF-TSS implementation, first with the project organisation, followed by five stages and four milestones, as used in Section 8.9.

10.7.1 ISD Project Organisation for PBSDF-TSS

The basic ISD project organisational structure is the same as Figure 1.1. The ISD project team (referred to as the project team for short) acts on behalf of the client as the principal, while software suppliers are the agents. The project team is responsible for specifying functional requirements and technical constraints, co-ordinating request for proposals (RFP), evaluating supplier proposals, managing requirements changes, conducting product acceptance evaluation and carrying out system implementation in the end. The key teams within the project may include a Requirements Management Team (RMT), a System Level Architecture Team (SLAT), a Supplier Management Team (SMT) and a Quality Assurance Team (QAT), among others (Figure 10.2).

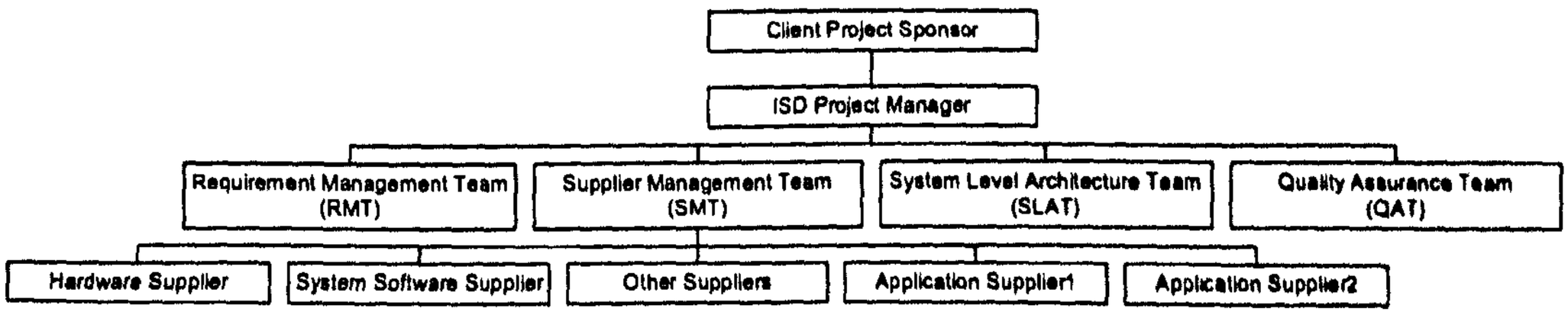


Figure 10.2: A typical ISD project organisation for implementing PBSDF-TSS

RMT's role is to co-ordinate with users, user managers and other teams within the client organisation to maintain a set of up to date requirements throughout the duration of the project. Each requirement must be numbered and traceable (Gilb, 1997). The RMT has to

tread a fine line between making clear what is required and how to meet the requirements. Each requirement change must be communicated to both suppliers at the same time through SMT. SMT registers all requirements, requirement changes and negotiates with suppliers regarding possible impact any change may have on the overall price and delivery time. All changes must be fully recorded so that at any moment of time, the client knows the projected cost and the actual cost to date for each supplier. In addition, the project team must also know at any point the likely total payment to suppliers based on different possible scenarios. Part of the SMT's responsibilities is to work with RMT and SLAT to question and record the assumptions that are made by suppliers on a comparative basis. Some of the assumptions, if accepted, may have to be converted to requirements and communicated to both suppliers. QAT is to carry out product evaluation in association with RMT at the evaluation stage and to provide support to other necessary test activities (e.g. usability test, performance test etc).

10.7.2 Stage 1: Client Preparing "Request for Proposal (RFP)"

The client first considers various software acquisition options available. When a client decides that an externally developed software application is required, an RFP is prepared to solicit proposals from suppliers. The RFP has to make clear to all suppliers that PBSDF is used to manage software development and two suppliers will be selected to participate in a development competition. In addition to necessary technical details and the usual projected total price, a supplier is asked to quote a Non-Acceptance Price Factor λ according to the definition given in Section 10.2. The client must take care in formulating RFP in such a way to make clear business needs without prescribing implementation details. In other words, the client should and should only state what is required and not how requirements are to be met. The requirements at this stage are not expected to be complete but should include as much as possible "revealed requirements" as defined in the Kano Model (Section 8.3).

10.7.3 Milestone 1: RFP Issued

Without having to worry about "freezing" the requirements, it is possible to issue RFP relatively soon after the project initiation. It is critical, however, for the client to control requirements strictly. If there is any revision to the requirements included in the RFP, each supplier must be notified at the same time with the same revision. Each supplier must submit its proposal based on the same version of requirements for ease of proposal evaluation and supplier selection.

10.7.4 Stage 2: Pre-Contract Proposals

Suppliers submit proposals indicating how software will be designed and developed, with the understanding that two suppliers will be selected. Crucially, each supplier quotes a Non-Acceptance Price Factor λ in addition to a full asking price P. Supplier selection should be carried out with usual criteria like financial soundness, technical capability, development capacity and value for money etc. One important feature of PBSDF is that the opportunistic pricing behaviour as a result of high switching cost is likely to be curtailed (see Section 10.9 below). Pricing low initially will help a supplier being selected, but charging higher prices later diminishes its competitiveness. What is more, suppliers do not monopolise development. Therefore any price change is subjected to competitive pressure.

10.7.5 Milestone 2: Client Committing to Contract with Two Suppliers

Having selected two suppliers, the client enters a PBSDF contract with each respectively. The essence of the contract is that the client will pay the supplier according to whether its product is accepted or not. This milestone is completed when the client gives the supplier a signal to proceed with development based on a clear understanding for P and λ and other relevant price factors. PBSDF-TSS provides a relatively complete contractual framework due to its consideration for various eventualities (Maskin, 2002). The question of product quality is left to the acceptance stage to resolve.

10.7.6 Stage 3: Software Product Development

Both suppliers are in a simple game: to develop a product cost-effectively that meets the client's business needs and technical constraints for it to be accepted. In view of the three types of requirements, the suppliers exert efforts so that the client's "revealed requirements" are met with no omission, the "expected requirements" are fully met and the "exciting requirements" are built into the product as much as possible. The objective is to delight the client with their products at the evaluation stage.

It is up to the suppliers to adopt the best practices and tools as appropriate. The end game is easy to understand for every one within the supplier teams. Each supplier team member should be incentivised to establish a close working relationship with the client throughout the development stage. The competitive pressure compels each supplier's development team to be more innovative and to be better supported by its own management team. In the process of development, questions may be raised, and changes may be made to the requirements specification. The changes are to be recorded, costed and incorporated into the requirements specification if adopted and communicated to both suppliers. Any price and delivery time variation is to be incorporated into the overall project cost model and project schedule. The competitive pressure will make sure that the price variations are checked. Any excessively high price on changes will make a supplier's product uncompetitive, *ceteris paribus*.

The management team within the supplier organisation has the incentive to provide adequate skilled resources for development if its product is to win the development competition. Developers are no longer just responding to written specifications. They are likely to be encouraged to be innovative and to think on behalf of users. They have to pay attention to non-functional requirements like performance, usability and reliability etc. In the face of continuing requirements amendments, suppliers have to make sure that their products are

easily adaptable, which helps to lower long-term product maintenance and enhancement costs for the client. Software designers, developers and testers within the supplier organisation work closely with one single objective: to have a system that will be accepted by the client.

10.7.7 Milestone 3: Suppliers Release Software for Evaluation

The key milestone is for each supplier to release its own developed product to the client at its committed delivery date. At the point of the official release, the product should be able to function as a "working product" (Section 10.3).

10.7.8 Stage 4: Client Conducts Product Evaluation and Considers Options

Upon receiving a software product, the project team installs the product onto a pre-prepared system platform. This stage traditionally has been called "Site Acceptance Testing (SAT)". In view of the new framework, it is more appropriately called "Site Evaluation Test (SET)". This stage should not be rushed. During evaluation, the project team takes advantage of any absolute measure available. More importantly, the technique of relative product evaluation should be used thoroughly to compare every conceivable product dimension between the two products. The following criteria might be considered, among others:

- Meeting functional requirements and technical constraints;
- Interoperating with legacy data, if applicable;
- Outputting data that meet corporate quality standards
- Supporting relevant business process;
- Compatible with corporate brand image;
- Being cost-effective to support;
- And possibly having a considerable number of additional "exciting" features.

The client evaluates each product's potential value contribution to its organisation. The evaluation outcome for each supplier may follow the scenarios described in Section 10.3.1.

10.7.9 Milestone 4: Client Announces Product Evaluation Result

The client communicates the result of product evaluation to each supplier. Necessary commercial transactions take place between the client and each of the suppliers according to the evaluation outcomes.

10.7.10 Stage 5: System Implementation and Post Implementation Activities

System implementation focuses on embedding the software system with client's infrastructure. The usual support arrangement is to be initiated at this stage. Application support teams are to be set up, trained and commissioned. Call centre and other support teams should be arranged as appropriate. The new information system should be fully supported while in operation.

10.8 Cost-Benefit Analysis of PBSDF-TSS

The implementation of PBSDF-TSS is similar to the idea of "build two, throw one away", which was suggested by Brooks (1975). But Brooks' approach refers to prototyping that has its fundamental problems (Section 3.7.2). PBSDF-TSS adopts a parallel strategy rather than a sequential one. Nevertheless, just like prototyping, there might be a concern with the cost of building two products. Is PBSDF-TSS cost-effective for a client? In this Section, the cost difference between using PBSDF-TSS and TSDF for software design and development is examined with the help of a simple total cost model.

10.8.1 Total Cost with TSDF

In TSDF, the total cost (C_0) of developing and supporting a software system for a client may be represented with the following equation:

$$C_0 = P_0 + \alpha P_0 + X_0 + Y_0 + Z_0 \qquad (10.1)$$

In which:

P_0: initial bid price by the supplier for design and development.

αP_0: the aggregate total of subsequent price variations after the initial accepted bid price.

The value of α is the percentage price variation compared with the initial bid price.

All price variations are included, e.g. requirements clarifications, requirements changes and contract re-negotiation etc.

X_0: aggregate total of ISD project cost items excluding those paid to the application software supplier. It includes costs for hardware and system software. It also includes costs for maintaining ISD project staff who charge their time to the project budget and costs for specialist services performed by other suppliers (project management consultancy, security audits etc.). X_0 may vary as a result of software delays or rework due to poor quality.

Y_0: aggregate total of costs that the client incurs during the ISD project but not charged to the project. Client senior managers often spend much time in overseeing troubled ISD projects. Such senior management time represents an opportunity cost to the client. A client may also assign to ISD projects critical internal resources paid for as the client's general overhead, rather than as part of project budget. There may be reduced company revenue or possibly regulatory fines as a result of late introduction of the intended software system. Like X_0, Y_0 may vary as a result of delays in software delivery or rework due to poor software quality.

Z_0: the aggregate total of costs related to a software system's operation, support and enhancement after its implementation. The client may have to increase the size of the support team if a software system crashes regularly and does not work with operational data as expected (e.g. The CAPSA case in Finkelstein, 2001). If only one third of the contracted deliverables is actually delivered, the client may have to pay further costs for either enhancing the developed system later or to start another project to develop the remaining functionality. Of course, the additional cost might be opportunity cost for having a poor system. All these costs are included in Z_0.

The value of α in (10.1) represents a supplier's percentage price increase compared with the accepted bid price. There may be many reasons for price increases in TSDF. For any particular product, α represents a supplier's propensity for price increase (PPI). Neither the client nor the supplier knows how high PPI is in the beginning of a project. Consider the MSS price history for example (see Table 4.3 in Chapter 4), the initial budgeted price for its software development was £240K on 23/06/2001. By the end of the project, the same part of the expenditure increased to £867K on 13/03/2002. The value of α in this case is 2.61. In other words, MSS went over its initial price by 261%. Part of the price increase came from a "re-planning" exercise and part from project delays.

10.8.2 Total Cost with PBSDF-TSS

With PBSDF-TSS, a similar total cost model can be constructed. Assume that Supplier A and Supplier B are selected to participate in the development competition. During the product development, there are a number of price changes. In the end, Supplier A's product is accepted and Supplier B's product is rejected, the total cost to the client is C_1:

$$C_1 = (1 + \beta_A)P_{1A} + \lambda_B(1+\beta_B)P_{1B} + X_1 + Y_1 + Z_1 \qquad (10.2)$$

In which:

P_{1A}: Supplier A's bid which is accepted to go forward to the full product development.

β_A: a percentage price change requested by Supplier A before the product is delivered.

λ_B: Non-Acceptance Price Factor quoted initially by Supplier B.

β_B: a percentage price change requested by Supplier B before the product is delivered.

P_{1B}: Supplier B's bid which is accepted to go forward to the full product development.

X_1, Y_1 and Z_1: These three items correspond with X_0, Y_0 and Z_0 in (10.1).

The first item $(1 + \beta_A)P_{1A}$ in (10.2) is the total price the client pays Supplier A for its software product that is accepted. The second item $\lambda_B(1+\beta_B)P_{1B}$ is the total price the client pays Supplier B for its software product that is rejected.

10.8.3 Comparing C_0 and C_1

The question to be examined is if $C_1 \leq C_0$. For the sake of simplicity, it is assumed that $P_{1A} = P_{1B} = P_1$. Further, it is assumed that both suppliers have quoted the same λ value and requested the same β percentage price variation during development competition. The delta between two total costs can then be expressed as:

$$C_1 - C_0 = ((1+\lambda)P_1 - P_0) + ((1+\lambda)\beta P_1 - \alpha P_0) + (X_1 - X_0) + (Y_1 - Y_0) + (Z_1 - Z_0) \quad (10.3)$$

Consider each item in turn:

$((1+\lambda)P_1 - P_0)$: If suppliers quote their full true prices before contract with or without PBSDF-TSS, then $P_1 = P_0 = P$. When that happens, the client pays λP more with PBSDF-TSS than otherwise. However, in the analysis of Section 8.4, it is pointed out that, with competitive bidding in TSDF, P_0 is quoted below the true price. In PBSDF-TSS, suppliers are likely to quote their true prices (Section 10.9). Therefore, this item is likely to be positive.

$((1+\lambda)\beta P_1 - \alpha P_0)$: This item can be either positive or negative depending on α, β and λ. If there is no price variation whatsoever (no re-planning, no change of requirements and no clarification of the requirements etc) then $\alpha = \beta = 0$ and this item is zero. If there are many price variations, the price change represented by αP_0 can be high due to monopoly pricing compared with βP_1 in PBSDF-TSS. Therefore, $\alpha > \beta$. The difference between β and α depends on the demand elasticity of the software product in question and the number of changes required during the project. Considering the peculiarity of software demand and possible lock-in situations (Varian, 2001), α is likely to be far greater than β. Overall, the item may be either positive or negative.

$(X_1 - X_0)$: This is likely to be negative. Cost saving is possible due to a better chance that the product will be delivered on time with better quality. Though dealing with two suppliers may be more demanding on the client, the extra cost may be compensated

by efficiency gain as a result of less waiting time on the part of the project team.

$(Y_1 - Y_0)$: This is also likely to be negative. On time software delivery with better quality will reduce the need for the client management involvement, thus reducing the opportunity cost for their time.

$(Z_1 - Z_0)$: The last item is likely to be negative since better quality software systems will cost less to operate and maintain. The client may also benefit from a more competitive support contract at the time of product evaluation.

The first two items in (10.3) represent the net supplier payment difference for acquiring the software product using the two different approaches. The two items can be written as Δ_s:

$$\Delta_s = ((1+\lambda)P_1 - P_0) + ((1+\lambda)\beta P_1 - \alpha P_0) = (1+\lambda)(1+\beta)P_1 - (1+\alpha)P_0 \qquad (10.4)$$

Δ_s may be modelled graphically. Figure 10.3 provides one possible scenario where $P_0 = 100$, $P_1 = 110$, $\beta = 0.5^*\alpha$ and λ varies from 0.1 to 0.4.

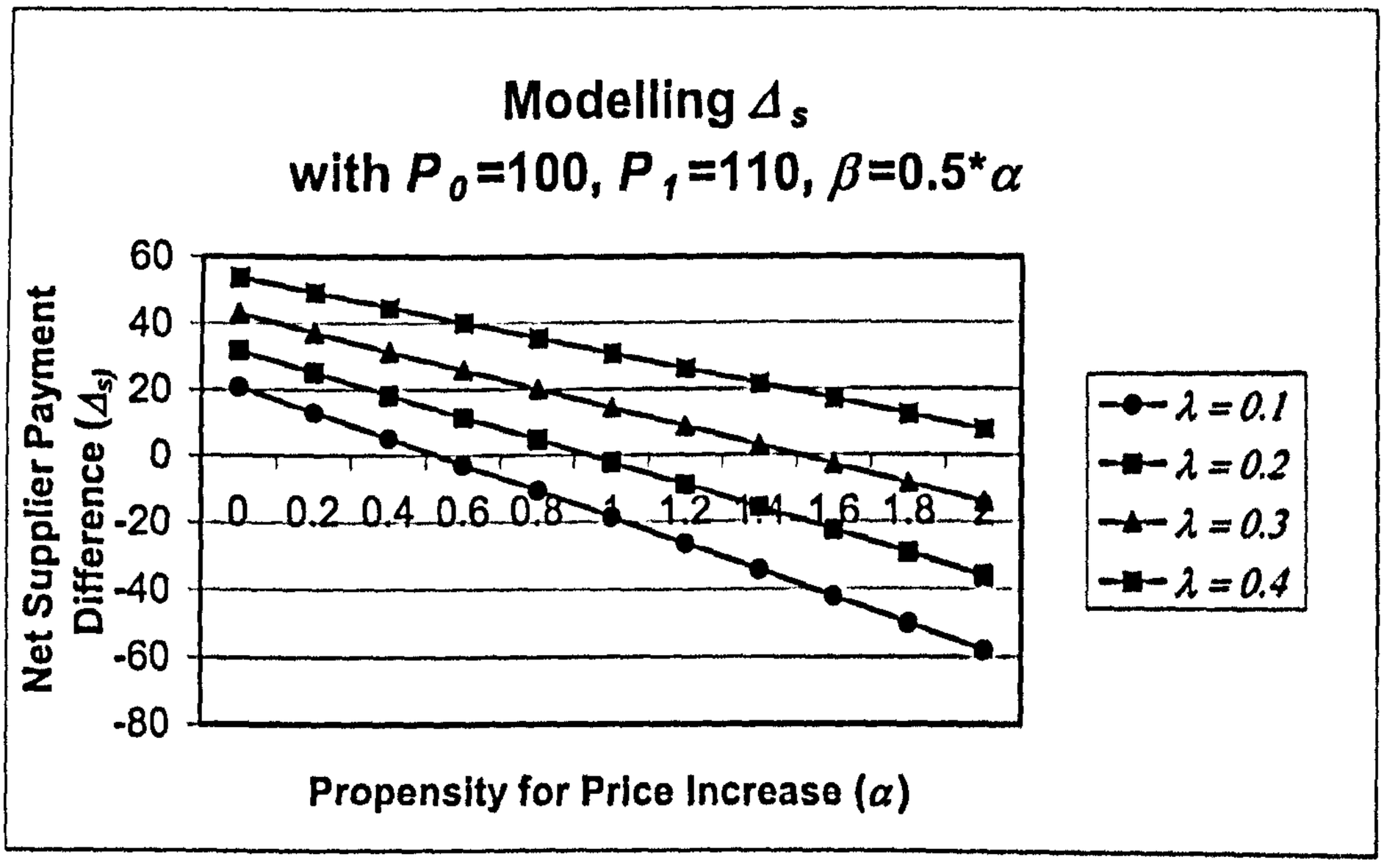


Figure 10.3: Direct payment difference for software product (Δ_s) between using PBSDF-TSS and TSDF

Figure 10.3 demonstrates that under the assumptions given regarding P_0, P_1, α and β, the total payment for software product in PBSDF-TSS may be less compared with that in TSDF when α is high (typically α in the region larger than 1.0) and λ is agreed at a relatively low level (e.g. $\lambda \leq 0.2$). Such outright cost saving is possible due to the competitive mechanism in PBSDF-TSS. When the possibility of price variation is moderate (e.g. α between 0.3 to 1), a client pays less than 40% extra if λ is agreed to be 0.3 or lower. It is only if there is little chance of price increase ($\alpha < 0.3$), combined with a high λ (e.g. $\lambda \geq 0.3$), when a client is likely to pay over 40% extra using PBSDF-TSS. If a supplier is able to deliver a software product within cost and schedule targets and with the expected quality (i.e. low uncertainty of the development outcome compared with the supplier's cost, schedule and product promises), using TSDF may cost less. If, on the other hand, there are many opportunities for price variations in developing a product, it may cost the client less to use PBSDF-TSS.

In developing and operating an information system, the direct payment to suppliers is only one part of the total cost. The remaining three cost items in (10.1) and (10.2) represent a significant proportion of the total cost of ownership. Consider Case 2 as an example, the cost for AES' software product is only 21% of the total project budget (Section 5.8.4). In Case 3, the software product cost is 17.1% of the total project cost (Section 6.6). Thus, there is much room for efficiency improvement in areas other than the payment for the software product. Any reduction to product delay and improvement to product quality are likely to reduce expenditure in other areas. In particular, software maintenance costs a great deal to client organisations. Boehm (1976) suggests that software maintenance cost is 40% of the total cost of owning information systems ("the overall hardware-software dollar" in Boehm's words). Boehm predicts that the maintenance cost percentage would have grown to 60% by 1985 and increased further thereafter. Further studies may be carried out to verify Boehm's prediction empirically. If it is accepted that software maintenance takes a significant proportion of the

total cost of owning a system, it implies that there would be a great potential for cost saving in this area. In any case, further studies may also be carried out to model the effect of PBSDF-TSS on cost items other than the direct payment to software suppliers. In the meantime, it is worthwhile to point out that the main benefits of PBSDF-TSS come from possible quality improvement and reduced development risks, not direct savings in software product cost.

10.9 Game Theoretic Modelling of PBSDF-TSS

Section 8.9 provides a game theoretic model for the client-supplier contracting process within TSDF and explains that the process involves "non-credible promises". In this Section, PBSDF-TSS is modelled using concepts and techniques from the game theory to see how the non-credible promises may be overcome. PBSDF-TSS is potentially complex to model as a whole. It is therefore broken down into three subgames. One is the contract bidding subgame played among suppliers (Section 10.9.1). The second is the subgame played between two competing suppliers in development competition (Section 10.9.2). The third is the contracting process between the client and the supplier (Section 10.9.3). Section 10.9.4 examines how the presence of more than one supplier affects suppliers' behaviours so that the "non-credible promises" are overcome and how PBSDF-TSS provides the client with superior value compared with TSDF.

10.9.1 Subgame for Contract Bidding

In game theory, bidding is studied as one of the auction games. There are auctions to buy and auctions to sell. Auctions for bidders to buy have been most often studied but the principles apply equally in reverse to auctions for bidders to sell (Gardner, 1995). When suppliers bid for contract, they are in an auction to sell their goods and services. In ISD projects, suppliers make their bids simultaneously with information hidden from each other; hence they are "sealed-bid auction to sell". In addition, if only one supplier is to be selected, as in TSDF, the rule of the game is usually that the bidder with the lowest price wins the contract, *ceteris*

paribus. This is called a "first-price auction". If the rule of the game is altered so that the lowest bidder is paid the second lowest price, the game is called a "second-price auction". The second-price auction is rare in practice but it has interesting features (Gardner, 1995) that are relevant to PBSDF-TSS.

By studying the "first-price auction" with complete information, Gardner (1995) concludes that bidders do not bid with their true evaluation. In an auction to buy, bidders underbid to increase the buyer's surplus. In an auction to sell, bidding above the true cost allows the supplier to pocket the seller' surplus. Such bidding behaviours are seen in the market exchange place where sellers ask for high prices, expecting to be haggled downwards. In TSDF, such a bidding pattern is affected by high switching cost in the software product market, which may have led to underbidding in ISD projects (Section 8.4).

In a second-price auction to buy, however, the bidding principle is that bidders "always bid true value" (Gardner, 1995). Gardner adds:

> "This principle holds true regardless of the number of bidders... Moreover, this principle does not depend on knowing your rival's valuation. It is just as true if you don't know your opponent's true value. Even in that case, paying the second-highest price means you can't lose when you bid true value." (p. 307)

If this principle is applied to the second-price auction to sell, the supplier should always bid according to its "true price". Would this principle apply to PBSDF-TSS where two lowest-priced suppliers are to be selected, *ceteris paribus*? The fact that a client is prepared to select two suppliers demonstrates that the client is willing to pay at least up to the second highest price. However, the game rules of PBSDF-TSS are significantly more complicated since:

- Two lowest priced suppliers with a set of bidding information (P_0, λ, μ etc, not just a single value) are selected to participate in the development competition;
- Both suppliers are allowed to adjust their total prices after the contract award to allow requirements changes (either scope increases or decreases);

- No adjustment of the price factors (λ, μ etc) is allowed;
- A supplier wins the development competition if its total price is lower, *ceteris paribus*;
- The winning supplier is paid its full asking price P while the other is paid its λP.

In PBSDF-TSS, two lowest-priced suppliers only win the rights to participate in the subsequent development competition rather than being paid directly at that point. The eventual payoffs depend on the outcome of the development competition. Due to the uncertainties associated with the cost estimation and actual development activities, PBSDF-TSS becomes a complicated game to model. No equivalent game theoretic model has been found in the literature (Elgood, 1984; Gibbons, 1992; Gardner, 1995; Bierman and Fernadez, 1998) to match the PBSDF-TSS game rules to provide a conclusive answer to the question whether suppliers' best strategy is to bid with their "true prices". Whether this is the case warrants further separate studies (Section 11.6.2). In the meantime, the effect of the switching cost on the bidding prices is modelled next.

10.9.2 Subgame for Development Competition: The Effect of Switching Cost

Whether the suppliers participating in the initial bidding will be proven to bid with their true costs, PBSDF-TSS is designed to overcome the possible opportunistic price increases due to high switching costs. Consider two suppliers, Supplier1 and Supplier2, who have won the initial bidding by submitting the same lowest price $P_{bidding}$ with different private valuations of the client's switching cost S_{high} and S_{low}. In other words, their undistorted prices should have been $P_1 = P_{bidding} + S_{high}$ and $P_2 = P_{bidding} + S_{low}$ respectively. Both suppliers try to recover their full prices through subsequent price increases. The outcome of the development competition can be modelled in the normal form shown in Table10.4. Both suppliers adopt the dominant strategy to deliver an acceptable product. When the products from both suppliers are acceptable, Table 10.4 shows that Supplier1 with the high price $(P_{bidding} + S_{high})$ will be only paid $\lambda_1(P_{bidding}+S_{high})$ while Supplier2 with the lower total price will be paid the full price

($P_{bidding} + S_{low}$). In other words, in PBSDF, it does not pay to bid low to win contracts, only to be paid a non-acceptance price if its final price is uncompetitive. This is in sharp contrast to the bidding in TSDF where suppliers compete to underbid and are then free to increase its price in the bargains-then-ripoffs opportunistic pattern.

Table 10.4: Modelling the effect of private switching costs in PBSDF-TSS

		Supplier2	
		To deliver an acceptable product	**Not to deliver an acceptable product**
Supplier1	**To deliver an acceptable product**	$(\lambda_1(P_{bidding}+S_{high}), (P_{bidding}+S_{low}))$	$((P_{bidding}+S_{high}), \lambda_2(P_{bidding}+S_{low}))$
	Not to deliver an acceptable product	$(\lambda_1(P_{bidding}+S_{high}), (P_{bidding}+S_{low}))$	$(\lambda_1(P_{bidding}+S_{high}), \lambda_2(P_{bidding}+S_{low}))$

10.9.3 The Contract Between a Client and a Supplier

The client-supplier PBSDF contract can be modelled as a dynamic game with complete information (Gibbons, 1992; Section 8.9.2). It is dynamic since each player's action is sequential to the other. It has complete information since the players are assumed to know each other's pay-off functions based on the pre-contract understandings. The game is shown in the extensive form (Figure10.4).

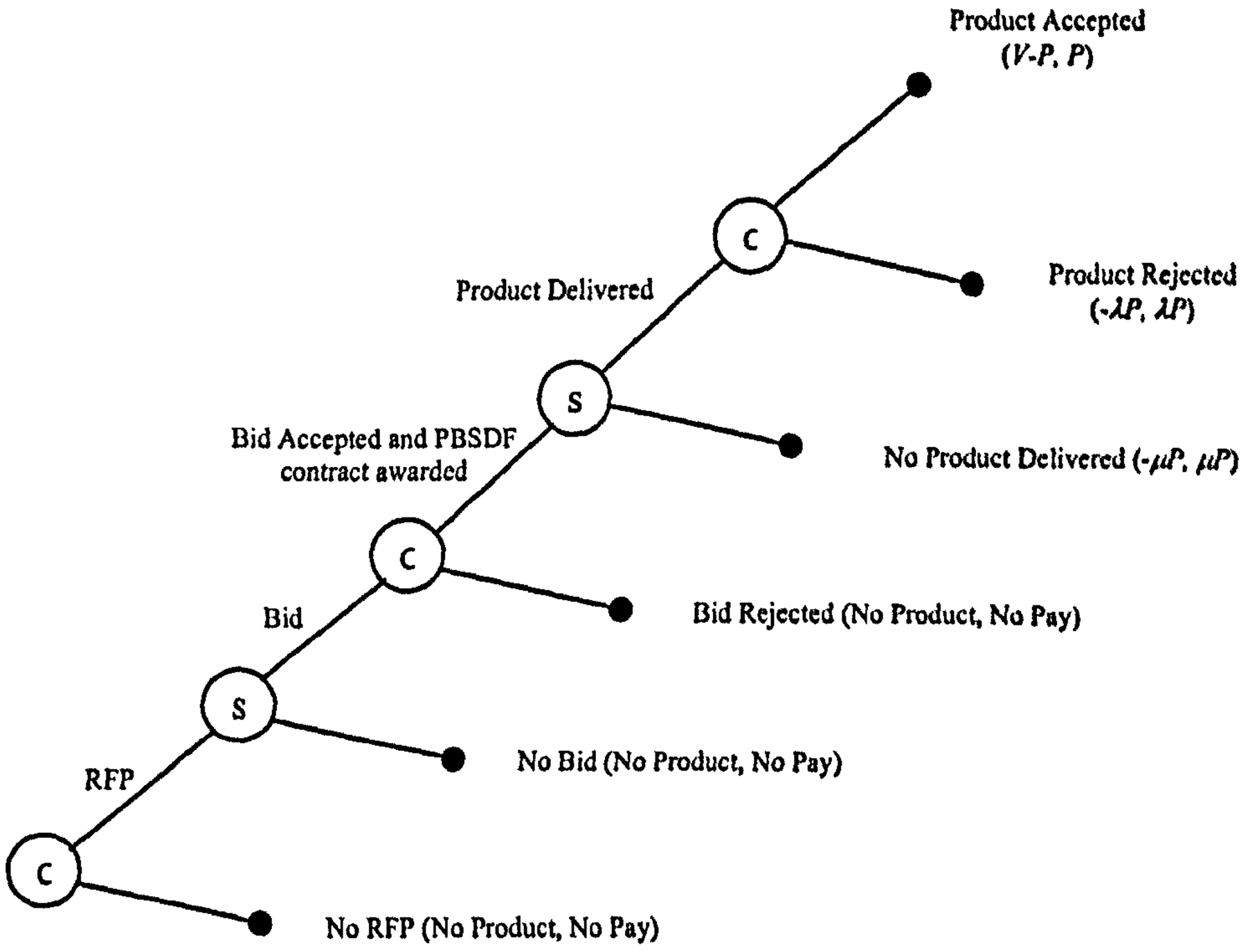


Figure 10.4: The game of client-supplier contract in PBSDF-TSS

In Figure 10.4, a client (C) decides to issue RFP for a software application to a supplier (S) or not. If the client decides to develop the software internally, it does not issue RFP. The game ends with no product from any supplier and the supplier is not paid anything. If an RFP is issued, the supplier must decide to bid for the contract or not. If it does not bid, the game ends for that supplier. If it submits a bid, the client decides to accept or reject the bid. If it rejects the bid, the game ends. If the client accepts the bid, the supplier is awarded a PBSDF contract. If the supplier then fails to deliver a product on a pre-agreed date and the contract is terminated as a result, the supplier is paid μP. The client loses μP and gets no product. If the supplier develops and delivers a product, the client evaluates it and decides to accept it or not. If the client rejects the product, the client pays the supplier λP and gets no product. Otherwise, the client pays P, accepts and deploys the product, enjoying the benefit V associated with having the product.

At the last step, the client gets the chance to assess the value of the delivered product. If there is only one supplier's product to consider, theoretically the condition for the client to accept the product is:

$$V - P > -\lambda P \qquad \text{or} \qquad V > (P - \lambda P)$$

This illustrates that a client would be willing to accept a product even if the product value is less than the price paid for by the margin of λP. This conclusion has two implications. Firstly, the value λ can make a difference between a client accepting a product or rejecting it. If λ approaches 1, a client may accept a product with a value approaching to 0. This happens in many software projects using TSDF. The implementation of poor quality software (e.g. CAPSA system in Finkelstein, 2001) is partly due to difficulties in evaluating the software benefits, but partly due to the fact that λ is set at 1 by default in TSDF. In that case, it is better for the client to have the product for paying the supplier than not having it at all. Secondly, if

for whatever reason, the client valuation of the software product is forced up by some external factors other than the intrinsic product value, a client may retain a product however poor its quality may be. Consider an organisation under government regulation. If the organisation must have a system in place by a deadline or face a fine of £10million, even if the software has zero operational value, the client is likely to accept the product for being seen to have met the regulatory requirement. This is because V = £10million for having a product if the product can pass the government audit in some way!

There is a critical difference between the game depicted in Figure 8.2 for TSDF and the one shown in Figure 10.4 for PBSDF-TSS. In Figure 8.2, the client-supplier game ends with the supplier taking the last step. If development uncertainty demands too much extra resources from the supplier to make good its initial promises, the supplier chooses to make less effort and deliver poorer quality products, making its initial promises "non-credible". In Figure 10.4, the client takes the last step to evaluate the suppliers' products based on their value contributions, not the fulfilment of the initial promises, thus overcoming the problem of any non-credible promise.

10.9.4 The Effect of Development Competition On An Individual Supplier

The above three sections look at the PBSDF-TSS game as three subgames. Though there is no co-operation between any of the suppliers, the PBSDF-TSS game affects individual supplier's behaviours profoundly. First of all, the bidding behaviour is significantly different from that in TSDF. In a normal "first-price" contract bidding process, the supplier's bid is affected by two considerations. The first is to bid with a higher than the "true" price so to increase the seller's surplus. This is overcome by PBSDF-TSS due to the selection of two suppliers, making the bidding analogous to the "second-price" auction (Section 10.9.1). The second consideration in supplier's bidding is the client's switching cost (Section 8.4). In TSDF, the higher the switching cost, the lower the supplier's bidding price, which is part of the bargains-

then-ripoffs behaviour pattern (Section 8.4). In PBSDF-TSS, the switching cost effect is removed with the help of a suitably agreed Non-Acceptance Price Factor (NAPF) λ and the development competition mechanism (Section 10.9.2). Incorporating these two effects, PBSDF-TSS induces suppliers to bid close to their "true" prices.

In addition to the improved bidding behaviour, PBSDF-TSS allows a considerable level of flexibility in terms of client requirements changes and encourages suppliers' development creativity to create value for the client. After all, it is the client value judgment that decides which supplier is the winner of the development competition. While Figure 10.4 describes the subgame between a client and one supplier, the decision to accept or reject a supplier's product depends on the evaluation of the other supplier's product at the same time. Therefore, the acceptance criteria for Supplier1's product is:

$$V_1 > (P_1 - \lambda_1 P_1) \qquad \text{and} \qquad V_1 > V_2$$

Similarly, the acceptance criteria for Supplier2's product is:

$$V_2 > (P_2 - \lambda_2 P_2) \qquad \text{and} \qquad V_2 > V_1$$

Both suppliers are in exactly the same position if they are treated fairly by the client. The above acceptance criteria suggest that suppliers will not only strive to deliver value for money to the client, but also will strive to deliver better value than the competing supplier. In this way, the competitive "invisible-hand" not only helps keeping the project cost under control, but also helps delivering product with functions, features and quality meeting or exceeding the client's value expectations. This is in stark contrast to TSDF where the supplier only aims to deliver with a budgeted level of effort, even though that means the client may be left with a product worth less than the paid price (Section 8.9.2). Of course, exactly how value can be measured is a separate subject that may deserve much further research (see Section 11.6.6).

10.10 Alternative Implementations of PBSDF

PBSDF-TSS may be suitable for a typical ISD project where a single client requires a single

instance of a software application. Development competition is necessary in such a case to manage development uncertainty and increase the likelihood that one supplier produces a product to be accepted. However, PBSDF may be applied with only one supplier without competition or with more than two suppliers in some circumstances, instances of which are discussed below.

10.10.1 Implementing PBSDF with a Single Supplier

Referred to as the Single Supplier Strategy (PBSDF-SSS), PBSDF-SSS requires a client and a single supplier to agree a PBSDF contract including a price and the associated price factors λ, μ and ν before the start of a project. The supplier is paid according to the development outcome. PBSDF-SSS may be useful if the client requirements are well defined (e.g. due to the availability of a legacy system). It may also be used if a client aims to develop an information system on a speculative basis and is prepared for not having a working software system in the end.

With only one supplier involved in the design and development, however, PBSDF-SSS is similar to TSDF, suffering from many limitations. First of all, divisions of responsibilities may be blurred between the client and the supplier. In using PBSDF-TSS, the client is responsible for specifying requirements. Both suppliers aim to satisfy stated requirements first to avoid having their products being rejected. As a result, both suppliers need a set of requirements that the client must provide. This is a desirable feature of PBSDF-TSS in that it facilitates a requirements-driven software development. With only one supplier, assumptions might be made by the supplier without being referred back to the client, which has led to delivery-led software development in TSDF (Section 7.4.2). Secondly, the single supplier is likely to quote monopoly prices for requirements changes. The client has no alternative source of information to verify prices and price assumptions. Thirdly, the client has no alternative system that can be used for relative performance evaluation. In addition, a client

may not have the luxury to reject the only product available. Even if a rejection decision is taken, it might be challenged by the supplier who may blame the client for a poor product. For all these reasons, PBSDF-SSS is not recommended.

One advantage PBSDF-SSS has over TSDF is that it provides an exit strategy for the client. If its limitations stated above are fully recognised, PBSDF-SSS may still be used in preference to TSDF. This may be the case if a client has a set of non-ambiguous requirements and a decision to accept and reject a product is easy to make. A fixed price and a fixed Non-Acceptance Price (instead of λ) may be agreed if there is little risk of later price variations. The client must be prepared to exit a project with a sunk cost and without a working system. Such a process is in fact operating in the market already. Two examples may serve to illustrate the mechanism. One was experienced by the author in SLC. A supplier specialised in data analysis offered a software tool to a client. The full product would have cost £150k for 10 users. But the supplier was willing to import the client's data and demonstrate to the client on one workstation for a trial phase for £10k to 15k. If the client did not like the product, either party could walk away with no further liability. In this case, the supplier quoted a full price and $\lambda \leq 0.1$. The other example is described by Collins (1998) about a £110million project for a "Customer System" that Andersen Consulting undertook for the Barclays Bank:

> "The Customer System was left mainly in the hands of Andersen Consulting on the simple basis that if its consultants failed to deliver they would not be paid and would never work for the bank again." (p. 291)

Collins does not provide any detail regarding the criteria of a "failure" to deliver. So far as the contractual arrangement is concerned, a literal interpretation is that $\lambda = 0$. These two examples might be viewed as possible ways of implementing PBSDF-SSS.

10.10.2 Implementing PBSDF with Multiple Suppliers

There are occasions when more than two suppliers are preferable to participate in the development competition, resulting in a Multiple Supplier Strategy (PBSDF-MSS). This is

especially the case where more than one instance of the software product is to be implemented. The key benefit for having more than two suppliers is that more than two software products will be available for evaluation, and thus more than one product may be accepted by the client. The rewards for suppliers can be linked into the eventual number of installations that each supplier secures. In that sense, PBSDF-MSS is similar to the development competition in the manufacturing industries where at least part, if not all, of the rewards may be from volume orders at the production (implementation) stage (Section 9.4).

PBSDF-MSS might be applied to large ISD projects with software applications developed centrally but implemented locally to many organisations. It may be particularly applicable in large government ISD projects where centralised data standards must be complied while local information needs may differ (Flowers, 1996). PBSDF-MSS will allow a number of feasible applications to emerge through partially funded development competition. Applications that are certified as to be compliant to central data standards (among other "revealed requirements") can offer choices to local institutions for adoption. If suppliers can work with local institutions closely, local variations may be catered for as part of "exciting requirements", so far these do not contradict the "revealed requirements". Resistance to adopting new systems has been a major problem in large government ISD projects with multiple client organisations (Southon et al., 1997; Flowers, 1996). PBSDF-MSS may help to overcome partially such resistance. Of course, much research may be carried out so that cost-benefit analysis can be conducted for PBSDF-MSS in more details, preferably with support of empirical case studies.

10.11 Discussions on PBSDF Implementation

Important to implementing PBSDF is the shared pre-contract understandings between the client and the suppliers as outlined in Section 10.2. Combining a complete contract with development competition and relative product evaluation, PBSDF provides a flexible

management framework. However, PBSDF is not designed to address all problems related to ISD projects. Neither does PBSDF recommend or restrict the use of any other ISDMs on the supplier's side. For example, being in the development competition, suppliers are free to choose either Structured System Analysis and Development Methodology (SSADM) or Object-Oriented Design Methodology (OODM) so far the suppliers can be confident of the benefits of any tools and techniques in helping them win development competition. In that sense, PBSDF addresses ISD project challenges at a high level. In this section, the possible use of PBSDF with some other techniques is explored.

10.11.1 Prototyping

Prototyping has long been considered as an important method of learning in software development. However, prototyping is associated with a series of problems when used on its own in ISD projects (Section 3.7.3). Within PBSDF, however, suppliers may find it helpful to construct prototypes to gain better understanding regarding functional requirements and technical feasibility. The client will also gain indirectly from a prototype by understanding a supplier's proposal better. Though building a prototype adds to project cost and time, the value to both sides could outweigh costs. Therefore, a funded prototype could be considered either before or after contracting a supplier. The CAPSA case (Finkelstein, 2001; Shattock, 2001) illustrates a missed opportunity in using prototypes:

> "An important step in the proposed process was for a direct test of both SAP and Oracle's solutions to take place in selected Departments. It was conceived that both suppliers would offer 'generic' systems making available the core functionality, and that these would be tested with Departmental data [...]. It is clear that neither SAP nor Oracle provided a demonstration system of the type envisaged. Oracle did give access to a remote system and staff in the Departments were able to see detailed demonstrations though not as far as I can determine with extensive University data. SAP, not unexpectedly, balked at the significant effort that providing a test system would entail and offered further demonstrations of industrial implementations of the SAP suite. Surprisingly the Departmental users of both systems gave the OK and both Oracle Financials and SAP moved to the next selection stage." (Finkelstein, 2001, §7.29)

The initial design of the user test was effectively asking suppliers for prototyped systems, and importantly, with "Departmental data". It was not clear why Oracle Financials did not construct the required prototype. It was clear that SAP did not provide such a prototype due to cost consideration. Had the client insisted on a prototype as initially planned with a reasonable amount of funding, the "unmitigated disaster" of CAPSA Project (Finkelstein, 2001) could well have been mitigated somewhat. The contribution of PBSDF is that it provides a funding mechanism that can cater for planned or unplanned prototyping needs.

10.11.2 Requirements Management

Poor requirements have been often blamed for failed ISD projects (Bronzite, 2000). Gilb (1997) suggests differentiating requirements when he states:

> "We need to articulate the difference between requirements that require design engineering, e.g., quality and cost-level improvements, and true application "functional requirements," which merely require implementation or acknowledgement in existing systems."

However, Gilb does not clarify the ownership of the different requirements. PBSDF adopts the Kano Model for requirements management. The client is responsible for specifying and controlling functional requirements while suppliers must manage non-functional and "exciting" requirements with technical expertise and innovation. Gilb does emphasise the need for "one legal instance" broken down into "elementary requirements" with identification "tags". This is made possible by PBSDF in that the client has to communicate to two or more suppliers at the same time. If the client controls requirements fairly and firmly, one legal instance is likely to emerge from the original requirements specification and subsequent changes. Of course, this assertion requires empirical testing (Section 11.5.2).

10.11.3 Quantified Product Evaluation

One remedy for product evaluation difficulties is the "quantified design evaluation" (Gilb, 1997). This still follows the absolute evaluation approach. It is costly, if not impossible to

build an absolute measurement model. A measurement model has the additional weakness in that it is only a model. Its feasibility could be open to dispute. However, should the quantified approach proves to be cost-effective, the client may apply it to all suppliers with the same quantified requirements and evaluation model. PBSDF enables the client to adopt relative product evaluation method *in addition to* making use of any absolute evaluation technique.

10.11.4 PBSDF for Large ISD Projects

PBSDF is applicable to managing software development in a relatively short time frame. This is due to the underlying assumption that a supplier has the incentive to participate so far its short-term variable cost is covered. Large projects tend to last longer and a supplier will not be able to cover its total cost over an extended period should its product be rejected. In other words, large projects carry high financial risks for participating suppliers. As a result, suppliers may increase their λ values accordingly. This could render a large project to be prohibitively expensive for the client using PBSDF. In such a case, a client may consider to break down a large project into smaller chunks with each developed using PBSDF. This is preferable since previous research results have shown that smaller projects are more likely to succeed (Standish Report, 1995; Johnson, 1999, 2000b). How a large project can be effectively broken into smaller ones is an important subject that should be studied by software system architects.

10.11.5 The Diversity of Software Systems

One question that might arise from the implementation of development competition is whether two systems developed by two suppliers will be more or less the same. If that were to be the case, the value of development competition as exploring different solutions would be lost. Design diversity is often discussed when consideration is given to designing and developing safety-critical systems (e.g. Inacio, 1998). In implementing PBSDF, the client has an opportunity at the contract-letting stage to screen the proposed systems. The client can

select the most promising proposals that are based on different approaches. While suppliers may pursue whatever solutions believed to be the best (both for the client and the supplier), the client may manage the risks by mixing conservative and innovative approaches using different suppliers. The resultant systems will then be different. The screening process may also prevent the selected suppliers to collude to develop a single system.

10.12 Summary

This chapter describes the outline of Performance-Based Systems Development Framework (PBSDF) as a possible solution to many of the ISD project challenges. First of all, a number of pre-contract understandings are outlined regarding requirements and the corresponding price and delivery time variations. Such changes are common in ISD projects and must be recognised and catered for. PBSDF is then outlined in three parts. Part I is a PBSDF contract based on project outcomes and corresponding payoffs. Part II uses development competition mechanism to reduce development risks and to encourage suppliers' innovation. Part III is a stage of product evaluation.

Three price factors are proposed as part of a PBSDF contract, namely Non-Acceptance Price Factor λ, Non-Delivery Price Factor μ and Development Cancellation Price Factor ν. The value of λ is critical to the workings of PBSDF, and may be influenced by the level of competition and other factors like the project length and risk levels of an ISD project. Section 10.4 examines the theoretical basis of setting the λ value. Section 10.5 provides empirical estimates from two types of data sources. Section 10.6 discusses a number of practical considerations suppliers might have in the process of setting λ.

Having outlined the basic understanding and concepts, Section 10.7 provides a step-by-step description of a typical implementation of PBSDF-TSS. The five-stage and four-milestone outline structure is used with details of how an ISD project might be organised and how

decisions might be taken by the client and suppliers. Section 10.8 provides a cost-benefit analysis for PBSDF-TSS using a total cost model from a client's viewpoint. Section 10.9 makes an attempt to model PBSDF-TSS as three subgames using concepts and techniques from the game theory. The first subgame concerns the contract bidding that takes place among all suppliers involved. The second subgame demonstrates that the switching cost effect is overcome in PBSDF-TSS. The third subgame illustrates how a client makes a product selection decision based on its own value judgement, regardless whether suppliers' promises are credible or not.

Variations of PBSDF using the Single Supplier Strategy (PBSDF-SSS) and Multiple Supplier Strategy (PBSDF-MSS) are described briefly in Section 10.10. Some issues in implementing PBSDF are discussed in Section 10.11. It is worth emphasising that PBSDF has been proposed based on the case studies and analyses in this thesis. It remains a "conjecture" until further empirical research can be carried out to test its validity, to refine its design and to apply it to reduce real world ISD project challenges and failures.

Chapter 11: Conclusions and Implications for Future Research

Truth emerges more readily from error than from confusion.
(Francis Bacon, quoted in Kuhn, 1970, p. 18)

11.1 Overview of the Study

The research journey for this thesis started when the author experienced first-hand symptoms of ISD project challenges. Puzzled in particular by difficult client-supplier relations and poor quality software products, the author set out to explore the underlying root causes and possible remedial actions. Three ISD projects experienced by the author provide the primary source of data for the research. Each project has been written up as an individual case with "within case" analysis examining the issues associated with each project (Chapters 4, 5 and 6). "Cross-case analysis" is then conducted to extract a common pattern of ISD project issues and challenges. The pattern matches and extends some of the existing descriptions of ISD project challenges found in the literature (Chapter 7). This is followed by an attempt to explain the challenges by identifying the root causes with reference to a number of concepts from the existing economic theories and other literature. The unified explanation suggests that ISD project challenges may have been caused by the Traditional Systems Development Framework (TSDF) characterised by a competitive-bidding-monopolised-development process (Chapter 8). This explanation implies the need for a competitive product development mechanism for overcoming the root causes associated with TSDF. The explanation is demonstrated to be a "warranted inference" through two contrastive analyses. Analogical analyses also suggest that the competitive mechanism is feasible with close client-supplier interaction (Chapter 9). A possible implementation of the competitive mechanism is outlined as the Performance-Based Systems Development Framework (PBSDF). A detailed cost-benefit analysis is conducted and possible varieties of PBSDF are discussed (Chapter 10).

11.2 Limitations of The Research

The overall research design for this study covers two stages of the research process (Stage 1 and Stage 2 in Figure 2.1). The other two stages of the same cycle, namely the stages of peer-reviews and application of research findings are not included according to the research strategy and research design (Section 2.4). These stages are left for possible future studies. In addition, there are other limitations as discussed below.

11.2.1 Limitations with Data Collection

This research is based on data collected through case studies based on participant observation. There are limitations associated with this method. First of all, the selection of the projects is limited to what was available to the author at the time of the study. The limitation must be recognised though overcoming it may require a number of further studies to be carried out using similar methods in different organisations. Secondly, the case studies rely on the author's interpretation of events and relations among the contracting parties. Though extensively checked by documentary evidence (Section 2.5), the general outlook is still constrained by the author's limited perspectives.

11.2.2 Limitations with Analysis

The analysis conducted in this thesis follows the IBE reasoning framework (Section 2.6). There are two stages to the IBE reasoning. The first is to formulate an initial hypothesis as the best potential explanation. In the second stage, the explanation is subjected to contrastive analyses and compared with other explanations. There are limitations with both stages. In the process of formulating the unified explanation, a number of concepts and theories are used (Chapter 8). These are limited by the author's personal knowledge about the existing concepts and theories in the management sciences. Other concepts and theories may be used to shed further light on the issues under discussions. During the contrastive analysis, selections of other contrasts are possible and may emphasise different causes.

11.3 Main Contributions

This section outlines the main contributions made by this study to the subject of managing application software suppliers in ISD projects. In the process of research, other relevant issues have been discussed, some of which are presented in Section 11.4.

11.3.1 Confirming Widespread Challenges in ISD Projects

Though the research has been based on three cases studies, they are representative of the population of ISD projects in many ways. The projects were "encountered" rather than "sampled". The projects involve different client organisations and different software suppliers. They were managed by different project managers with different styles and experiences. The projects built different systems with different technologies. Though there are constants throughout the three cases, for example, the author was employed by the same employer company, such constants do not explain the wide-ranging project challenges documented in the cases. Instead, what was observed in the case studies confirms reports over the last thirty years about "the software crisis" (Chapter 7).

11.3.2 Root Causes of ISD Project Challenges

The research has identified six root causes for ISD project challenges (Chapter 8). These are summarised below.

Root Cause 1: Misconception about Incomplete Requirements

The problem of incomplete requirements in ISD projects is widely recognised. But its root cause has often been attributed to poor requirements management on the client side. This study suggests that this is not the case. With the help of the Kano Model, this study suggests that the client expects technical functions to be provided by the supplier without having to specify for them. The client may not even be aware of potential "exciting" features. The root

cause with poor requirements management is likely to be the misconception that requirements can be complete and should be provided by the client. This study suggests that client should only be expected to provide and control "revealed requirements" (Section 8.3). The proposed management framework incorporates this conclusion (Section 10.2.2).

Root Cause 2: High Switching Cost

High switching cost in acquiring software has been recognised in the literature (Varian, 2001). The impact of the high switching cost on software supplier management in ISD projects, however, has not been fully appreciated in the past. One effect is that the high switching cost causes software suppliers' bargains-then-ripoffs opportunistic behaviour pattern. This explains some of the unexpected supplier price increases and budget overruns in ISD projects (Section 8.4). The problem of high switching cost suggests that a potential solution to ISD project challenges should aim to reduce it. Development competition with two or more suppliers is one possible way that can achieve this effect, and is therefore adopted in designing the proposed management framework (Section 10.3).

Root Cause 3: Misconception about Contract Strategy

Incomplete contracts have also been widely recognised in ISD projects. There appears to be a misconception that a complete contract is possible based on a complete requirements specification. Due to the problem of incomplete requirements, it is necessary to find an alternative contract strategy (Section 8.5). This study proposes an alternative way to obtain a relatively complete supplier contract by agreeing development outcomes and corresponding payoffs between the client and the supplier. This approach is incorporated in the proposed management framework (Section 10.3).

Root Cause 4: Inseparability of Design and Development

Software design and development activities are highly intertwined. At least at the current

level of modelling capability, it has not been possible to separate development from design completely. This is recognised by the proponents of iterative software development methodologies. The inseparability of design and development makes it a necessity to contract suppliers to design and develop at the same time. This allows the supplier a considerable level of freedom to formulate the solution that might help the supplier to deliver the software product at lower cost. This may have caused the delivery-led rather requirements-driven software product development observed in the case studies (Section 8.6).

Root Cause 5: Development Uncertainty

Development uncertainty refers to a situation when a developer knows what is to be achieved (the requirements), yet cannot be certain of satisfying the requirements by taking a particular course of actions. This leads to a trial-and-error approach of development. It may be overcome to a certain degree with a developer's increased experience and training but it cannot be overcome entirely. Development uncertainty cannot be easily controlled by planning or by imposing deadlines (Section 8.7). From a supplier's viewpoint, development uncertainty must be managed by balancing the need for more time and resources to explore different solutions against the need to be competitive in the market place. From the client's viewpoint, there is a need for a mechanism to stimulate supplier innovation yet keep the supplier cost under control. Development competition has the potential to achieve such a balanced outcome (Section 10.3.2).

Root Cause 6: Absolute Product Evaluation

In traditional ISD projects, a client contracts a software supplier based on its incomplete requirement specification. The requirements specification does not represent a complete product model with which the end product can be measured against. A software application is usually a highly complex product for the client to understand entirely. A software application, once integrated to be part of an information system, is also unique with its system platform,

network facilities and possible legacy data. The absolute product evaluation approach is difficult to apply without a complete design to be used as a quality benchmark. The suggested alternative is a relative product evaluation approach, supplemented by absolute measurements where possible (Section 10.3.3).

The six root causes interact closely in the traditional systems development framework (TSDF), causing the pattern of ISD project challenges. Some of the current solutions recognise some of the root causes and attempt to address them. In particular, the iterative approach recognises the problem of incomplete requirements and the inseparability of design and development. However, it does not address them adequately and it does recognise other root causes (Section 8.10). The unified explanation is that it is TSDF, characterised by a competitive-bidding-monopolised-development process, that underlies the ISD project challenges. This explanation is both consilient and simple (Section 8.11). It proved to be a better explanation than the existing one that claims "software is different" (Chapter 9). It also provides the basis of the proposed Performance-Based Systems Development Framework (PBSDF) in Chapter 10.

11.3.3 PBSDF - A Proposed Solution

To address the above root causes, PBSDF is proposed as a possible solution that incorporates the competitive development mechanism (Chapter 10). PBSDF can be described in three parts. The first part is to devise an outcome-based contract by mapping development outcomes to corresponding payoffs. The second part is a phase of development competition. The third part is a phase of product evaluation.

PBSDF Part I: A Outcome-Based Contract Between Client and Supplier

The simplest arrangement can be described as follows. At the contract bidding stage of an ISD project, each supplier is required to submit a normal full price P and also a Non-

Acceptance Price Factor (NAPF), designated as λ ($0 \leq \lambda \leq 1$). If a supplier is selected to participate in software development, the client and the supplier signs a PBSDF contract. The contract states that if at the end of the ISD project, the supplier's software product is accepted and deployed, the client pays the supplier the full price. If at the end of the ISD project, the supplier's product is not accepted, the client pays λP.

In the same way, a number of other project outcomes may be envisaged and their payoffs agreed between the client and the supplier. The more complete the set of outcome scenarios described and agreed in the contract, the more complete the resultant contract. The contract is based on a client requirements specification but does not depend on complete requirements. The supplier factors in the assumed risks in its commitment by setting λ competitively against other suppliers. This part of the management framework partly addresses the problem of incomplete requirements (Root Cause 1) and partly addresses the problem of contracting strategy (Root Cause 3) in TSDF.

PBSDF Part II: Development Competition

This part of PBSDF suggests the use of more than one supplier in developing a software product directly competing with each other based on the same set of client requirements. Development competition first of all addresses the problem of incomplete client requirements (Root Cause 1). This is because suppliers will compete to be innovative to satisfy the expected requirements and the unknown requirements in addition to the "revealed requirements" according to the Kano Model. Development competition also addresses the problem of high switching cost (Root Cause 2). Having more than one supplier implies that the client's switching cost in the duration of the ISD project is close to zero. This encourages suppliers to bid close to their true prices (Section 10.9). Thirdly, development competition recognises the inseparability of design and development (Root Cause 4) and encourages the iterative development within commercial constraints. Likewise, development competition

recognises development uncertainty (Root Cause 5). It encourages suppliers to try different solutions and to be innovative in delivering the end products.

PBSDF Part III: Combined Absolute and Relative Production Evaluation

With the suppliers' delivered products at the end of the development competition, the client engages in a product evaluation exercise using both absolute and relative evaluation methods. The objective of the evaluation is to decide whether a product can provide value to the organisation, not whether the product matches the original specification. When there is more than one product to be evaluated, the client has to decide which one provides more value. Therefore, the client focuses on the value a product can provide. The client may reject a product if it appears to match the client's requirements specification but is perceived not to provide value for money, or not to provide more value for money than the competing product(s). With the relative product evaluation method, the client is also able to evaluate dimensions not easily measurable in absolute terms (e.g. usability). This part of PBSDF addresses the problem of development uncertainty (Root Cause 5) in that potential failures are anticipated. It also addresses the pitfalls of absolute product evaluation (Root Cause 6).

11.3.4 Applicability of PBSDF to Different Types of ISD Projects

PBSDF as a whole addresses the six root causes of ISD project challenges as outlined in Section 11.3.2. In implementing PBSDF, there is a considerable degree of flexibility. The client may choose from different implementations of PBSDF including Single Supplier Strategy (PBSDF-SSS), Two Supplier Strategy (PBSDF-TSS) and Multiple Supplier Strategy (PBSDF-MSS). PBSDF-SSS may be useful if client requirements are well defined without much ambiguity. It may also be useful when a client undertakes speculative ISD projects and is prepared to have no working information system at the end of the project (Section 10.10.1). PBSDF-TSS may be applied to ISD projects where there is a single target system installation and requirements are not complete. PBSDF-MSS may be applied to large ISD projects where

there are multiple target installations. Each of the target installations might have slightly different informational needs (Section 10.10.2).

PBSDF is essentially a contractual framework that can supplement the existing project methodologies like Prince 2, which does not address the issue of contract management (CCTA, 1997). So far as the supplier is concerned, working within PBSDF means that the supplier is free to use any software development methodology. The single objective for the supplier is to develop a software product that can bring as much value to the client as possible to win the development competition.

11.4 Other Contributions of this Research

This study makes a number of other contributions in addition to the exploration of root causes of ISD projects challenges and the proposition of PBSDF. Some of the main ones are summarised below.

11.4.1 Project Success and Failure Criteria

In contrast to the traditional "Golden Triangle" approach to measure the success or failure of a project, this study suggests a two-tier approach considering the successes of the product separate from that of the project. A project may be able to produce a highly successful product yet the project execution may fail on the cost and schedule targets. On the other hand, a project may fail to produce a product but does not exceed the cost and schedule targets. This study suggests a scheme of "Qualified Project Success" and "Controlled Project Termination" to describe the two scenarios (Section 3.3).

11.4.2 Contribution to Project Management Research

Project management research has been biased toward practicality (Morris, 2000). While this study is firmly anchored in practical project experience, it is also based on concepts from a

number of theories including the agency theory, contract theory and game theory etc. These theoretical concepts help elucidating the complex client-supplier relationship and the client-supplier contract process. In the meantime, by using the technique of contrastive analysis, this study has made a distinction between one project type where design can be separated from development (e.g. construction projects and manufacturing product design projects) and another where design is inseparable from development (e.g. software development). The implication of making this distinction is that the project management approach and techniques for the first type cannot be transferred to the second one without carefully considering the suitability in the first place. In addition, the study casts doubt on the design-build contract management approach that has been advocated in the construction industry (Section 9.3).

11.4.3 Project Risk Management

Risk management for ISD projects has received much attention (e.g. Remenyi, 1999) but this study shows that risk management by identifying risks and making so-called provisions is not adequate. Risk management may be viewed both at the contractual and operational levels. First of all, there is the contractual risk that should be shared between the client and the supplier equitably. The T&M contract type is widely seen as a poor risk management mechanism (Morris, 1997). In contrast, the fixed price contract has been hailed as a performance-based contract type (e.g. Johnson, 2000b; Lichtenstein, 2002), which has been demonstrated not to be the case in this study (Section 8.5). On the other hand, simply shifting all risks to suppliers is equally unworkable (Morris, 1997). This has been somewhat borne out by unsatisfactory results from ISD projects using the PFI contract type (e.g. Arnott, 2003c). On the project operational level, a client must have at least one feasible option when risks become issues, whether planned for or not. Here the term "feasible" could be in terms of cost, time, product functions, product quality and organisational politics etc. The problem with TSDF is that many planned "mitigating actions" are not real options considering cost and organisational implications.

PBSDF provides a practical risk management mechanism. A number of price factors are to be agreed between the client and the supplier at the contracting stage. The most important one is the Non-Acceptance Price Factor (NAPF), which may incorporate a series of practical considerations (Section 10.6). For each risk, the worst option is envisaged in the contractual framework, which is not accepting a supplier's product and paying the supplier an agreed proportion of the full price. Through the use of price factors and development competition, PBSDF provides a flexible risk management framework.

11.5 Main Areas for Future Studies

This research is exploratory in attempting to resolve some of ISD project challenges. However, this is merely the beginning of a possible new research programme (Lakotas, 1978). Following the research process described in Section 2.3.2, much further work might be done, including research results dissemination and applying research results in real world ISD projects. In addition, new concepts and techniques may be used to analyse ISD projects. This Section looks at the main areas for future studies, limiting to the scope of ISD projects. Section 10.6 considers other potential studies that might be pursued further.

11.5.1 Disseminating the Research Findings

In addition to this thesis, a number of other channels may be used to disseminate the research findings, including publishing the research findings in academic journals and presenting the findings in conferences etc. An additional channel is to disseminate the research findings to organisations through management consultancy services. This has been started in a limited way by the author in having established a management consultancy company since April 2003. A number of presentations have been made to management consultants, senior management teams in both public and private sectors. There has been positive feedback. Some senior client managers are enthusiastic about PBSDF as a possible solution to ISD

project challenges. One management consultant has commented that, though his initial reaction was somewhat sceptical, the more he had thought about PBSDF, the more likely he considered it workable. Of course, there has been negative feedback as well. One attendant from the public sector commented during a presentation that PBSDF would not work, but was unable to explain why. Such peer-reviews contribute to the research process (Section 2.3.2). The author intends to continue the dissemination process and to engage a much wider audience in the debate of the strengths and weaknesses of PBSDF and the underlying assumptions and analyses.

11.5.2 Action Research Based Case Studies

This study relies on participant observation to collect first-hand project case materials. The research method allows the researcher to gain an insider view of ISD projects and to attain a contextual understanding of the contracting parties, project team members and their complex relationships. Such a method, though suffering from some limitations (Section 11.2), enables the researcher to assign meanings to events and documentary evidence in complex social organisations like ISD projects for in-depth analyses.

PBSDF is proposed as a possible solution to ISD project challenges. One possible strategy to further the research is to apply PBSDF in real world projects as part of action research studies. Kock (1997) gives a concise definition of action research as:

> "A general term to refer to research methodologies and projects where the researcher(s) tries to directly improve the participating organisation(s) and, at the same time, to generate scientific knowledge."

By applying PBSDF to real world ISD projects, a researcher will be attempting to achieve both aims. Such action research studies may be carried out by anyone who gains a good understanding of PBSDF and the underlying reasoning that has led to its formulation. PBSDF-TSS may be used for typical ISD projects carried out in commercial companies while PBSDF-MSS may be used in nation-wide government ISD projects (Section 10.10).

11.5.3 Adopting PBSDF for Internal Software Development

The software development "price index" for "own-account development", i.e. in-house development, shows a similar increasing pattern as that for CBSW and COTS in ISD projects (Section 9.2). There are similar problems with in-house ISD projects in terms of budget overruns and poor quality. Though managing internal software development may be different in many ways, the current in-house development process is essentially a monopolised one. Therefore, it is possible to apply the competitive mechanism to reform the development process. Internal competition may provide flexibility, stimulate innovation and motivate staff if properly managed (Birkinshaw, 2001). In addition, instead of competing at the product level, competition might also be held at a software component level. Two developers may be working to produce the same software component for selection. Such an internal development competition would require two developers working together both on a co-operative and competitive basis. It is worthwhile to compare such an internal development competition with the "pair-programming" technique advocated by XP (Beck, 2000):

> "All production code is written with two people looking at one machine, with one keyboard and one mouse." (p. 58)

Pair-programming has its advocates (Williams et al., 2000; Cockburn and Williams, 2000; Jensen, 2003) and critics (Rosenberg and Scott, 1997; Stephens, 2003). While pair-programming allows only one approach to be tried, the internal development competition allows two approaches to be experimented at the same time, overcoming at least partially the problem of development uncertainty. Internal development competition would have a profound impact on how an internal ISD project is structured and resourced. The impact assessment and cost-benefit analysis should be carried out separately as a research topic.

11.6 Other Areas for Future Studies

This study deals with supplier management in ISD projects that are related to many aspects of

software system development. It is not possible to list every single suggestion made as a result of this study regarding future possible research topics. This section selects a few topics that might be pursued in addition to the major areas discussed above.

11.6.1 Project Success Criteria

This study has proposed a two-tier success criteria approach that a project should be judged based on cost, schedule and product, and a product should be judged in turn by its functions, features and their corresponding quality dimensions (Section 3.3). This approach is suitable for projects that aim to produce products (including services). It would be interesting to follow up with a survey of project practitioners' opinions on such a scheme. Questions might include:

- How often, if ever, are projects undertaken not to produce a product or a service?
- If these projects do exist, what results are achieved to warrant these projects to be judged as successful?
- How would a controlled project termination (Section 3.3.3) be viewed in terms of project success or failure?

The answers to the first two questions might help to improve the definition of the term "project" (PMI, 2000). The last question deals with the perceptions of the suggested "controlled project termination". This might contribute to the study of project abandonment (Sauer, 1993; Keil, 2000; Royer, 2003).

11.6.2 Less Distorted Bidding Prices in PBSDF

An assumption in formulating PBSDF is that if two suppliers are to be selected in a contract bidding, suppliers will submit prices closer to their true prices (Section 10.9). This assumption is based on the conclusion in the game theory that in a second-price auction, bidders will bid with true prices (Gardner, 1995). However, in PBSDF, the winning bidders are required to compete with each other to win the final prize of supplying a software product

at the full price. This is different from a normal auction where a winning bidder enjoys the outcome immediately after the auction. A study may be carried out to prove or refute that the bidders will indeed be bidding with less distorted prices or true prices in PBSDF.

11.6.3 Realistic Project Feasibility Assessment

In a study highlighted by Glass (1998), feasibility study is one of the few things organisations like to do better before embarking on further ISD projects. The research for this thesis helps to understand why feasibility studies for ISD projects may not have been properly conducted in the past. Consider the impact of the high switching cost on supplier bidding prices. Suppliers bid low to win contracts and then charge higher prices later when opportunities are available, demonstrating a bargains-then-ripoffs opportunistic behaviour pattern. The initial low prices are taken into account by clients for feasibility studies. The net cost-benefit analysis will thus reflect a more favourable balance in undertaking certain projects than otherwise.

PBSDF is likely to force a number of adjustments in the price signals from suppliers to the client. First of all, suppliers are likely to bid with more realistic prices (Section 10.9), which will be higher than the opportunistically low bid prices in TSDF. Secondly, the risk of supplier development failure is made specific. Such a risk has too often been neglected in ISD projects when contracting parties share "enthusiasm and optimism" (Yourdon, 2002) in the beginning of projects. The risk premium is for the client to pay a certain extra amount. The Non-Acceptance Price Factor λ is built in with a number of other considerations like the supplier's opportunity cost. The net effect is that the client will have a more realistic cost-benefit analysis. The exact relationship between various risk considerations and project feasibility decisions may be modelled mathematically with further studies.

11.6.4 Modelling ISD Projects with Systems Dynamics

Research may be undertaken to model the effects of PBSDF on ISD projects using Systems Dynamics (SD). SD has been suggested as an effective tool to model business problems for experimentation. Sterman (2000) outlines the basics of systems dynamics and encourages a model-experimental approach to test solutions that might otherwise be elusive to our limited ability to reason. In fact, a number of attempts have already been made to model ISD projects with SD concepts and techniques (see e.g. Abdel-Hamid and Madnick, 1991; and more recently, Ruiz et al., 2001). SD may be used to model specific parameters in ISD projects like cost estimate (e.g. Ruiz et al., 2001) or at a strategic level for a whole IT division (Williford and Chang, 1999). Its application to project level modelling seems to be appropriate (Rodrigues and Bowers, 1996; Rodrigues and Williams, 1997, 1998). However, the assumptions made up to now in modelling ISD projects might benefit from re-examinations in view of this research study. For example, in modelling software development productivity, Abdel-Hamid and Madnick (1991) adopt the following model (based on Steiner, 1966):

> "Actual Productivity = Potential Productivity
> - Losses Due to Faulty Process" (p. 79)

Steiner's original model was for discussing group size and potential group productivity. The concern here is regarding the assumption that the same productivity model applies to software development projects. The validity of this assumption was not discussed in Abdel-Hamid and Madnick (1991). According to the research conducted for this thesis, software development is a distinctly different process compared with manufacturing and construction operations. Steiner's model may not be suitable for modelling software productivity since some loss of software productivity may be due to the inherent trial-and-error approach. The trial-and-error approach is a necessary part of software development and cannot be considered as a faulty process. On this basis, Abdel-Hamid and Madnick's SD model on productivity may be revisited. Existing SD models may be modified or new SD models may be constructed based on the new perspectives offered by this study.

11.6.5 The Interaction of Process and Data

Business process re-engineering (BPR) has not been particularly successful with "70 percent of all BPR initiatives fail" (Burke and Peppard, 1995). BPR has its origin in research within the IT field (Burke and Peppard, 1995). According to some researchers (e.g. Davenport, quoted in Galliers, 1995), IT plays a central role in BPR. The low success rate of BPR initiatives has raised questions about IT's role (Galliers, 1995). Should IT be maintained as the focus in BPR and or should IT be given a secondary role? This is a question that has been hotly debated in the BPR literature. However, "IT" has been often used as an abstract term at a high level without being precisely defined and it is not clear how it does or does not play a central role (e.g. Jones, 1995; Galliers, 1995).

As discussed in Section 3.2, business processes involving an information system straddle across users and software applications, often *linked* by data. Though users enter and manipulate data using software applications based on business processes, it is the data that records business transactions. Such transactions can only be properly interpreted with the understanding of the original business processes. In that sense, business data embody business processes. The interactions of users, processes, software applications and business data are complex. Failure to understand these interactions may help explaining some of the "contradictions" experienced in BPR (Jones, 1995). Research at the micro-level is required to understand how the interactions might have affected BPR programmes in the past. Such understandings may then help to guide BPR programmes in the future. It is hoped that BPR research can be steered away from the polarising debate between "IT-led" and "process-led" (Jones, 1995) to be re-focused on how to design and use information systems to implement desired business processes. This would be a significant undertaking and appropriate for separate research projects.

11.6.6 Value Management and Value Engineering

In surveying potential research areas within project management field, Morris (2000) points out an important research aspect to be "how project management can best contribute to improved business performance". Morris concludes that one of the research gaps lies in value management (VM) and value engineering (VE) in projects. "VM and VE have significant potential to business benefits; neither is well practiced outside construction" (Morris, 2000, p. 99). The assessment is probably particularly relevant to ISD project management. Value might be loosely defined as the net total of all benefits minus all costs. Part of the problem is that costs are difficult to establish for information systems (Bannister et al., 2001) and benefits perhaps more so (Mylonopoulos et al., 1995). PBSDF, however, affords the client a product evaluation and selection mechanism that allows the selection to be based on the perceived value contribution of the supplied software products (Section 10.3). PBSDF provides a crucial link upon which to develop a practical model to assist the product acceptance/rejection decision process. The construction of such a model requires much further studies of the costs and benefits that can be attributed to the application software in information systems.

11.7 Summary

This chapter outlines the research activities undertaken for the entire thesis and highlights some of the limitations of the research. It restates the research conclusions by summarising the problem areas in managing application software suppliers. It also outlines the proposed management solution, PBSDF, in three parts. The identification of the root causes and the proposed management framework are the main contributions of this research. A number of other contributions are also described.

A number of future potential research areas are briefly explored. Following the spiral research process discussed in Section 2.3.2, the main future areas are disseminating the research

findings for peer-reviews and applying PBSDF in action research studies. In addition, the possible use of the development competition technique for in-house software development is suggested. Other areas of future studies include further refining project success criteria, verifying the reduction of bidding price distortion in PBSDF, modelling ISD projects with System Dynamics using the new perspectives from the research and researching into the interaction of business processes and business data within the context of BPR. Since this study is exploratory in nature, it is apt to state that the research presented in this thesis might be the beginning of a new research programme on the subject of managing application software suppliers in ISD projects.

References

4TRESS (2003) *Product Architecture*. Accessed on 27/06/2003 at: http://www.aspacesolutions.com/pdf/4Tress%20Architecture%20Overview.pdf.

Abdel-Hamid, T. and Madnick, S. E. (1991) *Software Project Dynamics: An Integrated Approach*. Prentice Hall.

Abernathy, W. J. and Rosenbloom, R.S. (1969) Parallel strategies in development projects. *Management Science*, vol. 15, no. 10, pp. 486-505.

Abrahamsson, P., Salo, O., Ronkainen, J. and Warsta, J. (2002) *Agile Software Development Methods: Review and Analysis*. Accessed on 21/01/2003 at: http://www.inf.vtt.fi/pdf/publications/2002/P478.pdf.

Ackerman, K. B. (1996) Pitfalls in logistics partnerships. *International Journal of Physical Distribution & Logistics Management*, vol. 26 no. 3, 1996, pp. 35-37.

Adler, P. A. and Adler, P. (1987) *Membership roles in Field Research*. Sage Publications.

Adolph, S. (2000) *Lost in Chaos: Chronology of a Failure*. Accessed on 21/04/2003 at: http://www.sdmagazine.com/documents/sdm0001b (registration required).

Anderhub, V., Gächter, S. and Königstein, M. (2000) *Efficient Contracting and Fair Play in a Simple Principal-Agent Experiment*. Accessed on 27/12/2002 at: http://www.iew.unizh.ch/wp/iewwp018.pdf.

Aniceto, J. (2003) *Managing Scope Creep*. Accessed on 01/09/2003 at: http://www.suite101.com/article.cfm/17106/99319.

Anonymous (2001) *An immensely useful book!*, (book review dated 28/12/2001 on "Troubled IT Projects" by J. M. Smith). Accessed on 03/07/2002 at: http://www.amazon.co.uk/exec/obidos/ASIN/0852961049/qid%3D1061390645/202-5699895-8166254.

Anthes, G. H. (1994) *Air traffic takes another turn*. Accessed on 01/09/2002 at: http://www.computerworld.com/cwi/.

Apache (undated) *Apache HTTP Server Project Guidelines and Voting Rules*. Accessed on 27/02/2003 at: http://httpd.apache.org/dev/guidelines.html.

Arnott, S. (2003a) *Suppliers start race for £5bn NHS deals*. Accessed on 23/06/2003 at: http://www.computing.co.uk/News/1137897.

Arnott, S. (2003b) *How will future IT disasters be avoided*. Accessed on 08/05/2003 at: http://vnunet.com/News/1139433.

Arnott, S. (2003c) *Treasury rules out PFI for IT*. Accessed on 29/10/2003 at: http://www.accountancyage.com/News/1134235.

Arrow, K. J. (1986) Rationality of Self and Others in an Economic System. *The Journal of Business*, vol. 59, no. 4 (Part 2), pp. S385-S399.

Arrow, K. J. (1991) The Economics of Agency. In John W. Pratt and Richard J. Zheckhauser (eds.), *Principals and Agents: The Structure of Business*. Harvard Business School Press.

Avison, D. E. and Fitzgerald, G. (1988) *Information Systems Development: Methodologies, Techniques and Tools*. 1st Edition. Blackwell Scientific Publications.

Avison, D. E. and Fitzgerald, G. (1995) *Information Systems Development: Methodologies, Techniques and Tools*. 2nd edition. Blackwell Scientific Publications.

Avison, D.E. and Fitzgerald, G. (2003) *Information Systems Development: Methodologies, Techniques and Tools*. 3rd Edition. Blackwell Scientific Publications.

Babchuk, W.A. (1996) *Glaser Or Strauss?: Grounded Theory And Adult Education*. Accessed on 31/5/2002 at: http://www.anrecs.msu.edu/research/gradpr96.htm.

Bach, J. (1995) *The Challenge of Good Enough Software*. Accessed on 17/06/2003 at: http://www.satisfice.com/articles/gooden2.pdf.

Bach, J. (1997) *Good Enough Quality: Beyond the Buzzword*. Accessed on 17/06/2003 at: http://www.satisfice.com/articles/good_enough_quality.pdf.

Bakker, G. and Clark, L. (1988) *Explanation: An Introduction to the Philosophy of Science*. Mayfield Publishing Company.

Bannister, F., McCabe, P. and Remenyi, D. (2001) How much did we really pay for that? The Awkward Problem of Information Technology Costs. *The Electrical Journal of Information Systems Evaluation*. vol. 5, no. 1. Accessed on 19/11/2002 at: http://www.iteva.rug.nl/ejise/vol5/Ban-Mcabe_Ejitet.htm.

Beath, C. M., and Orlikowski, W. J. (1994) The contradictory structure of systems development methodologies: deconstructing the IS-user relationship in information engineering. *Information Systems Research*, vol. 5, no. 4, pp. 350-377.

Beck, K. (2000) *Extreme Programming Explained: Embrace Change*. Addison-Wesley.

Beck, K., Grenning, J., Martin, R. C. et al. (2001) *Manifesto for Agile Software Development*. Accessed on 19/12/2002 at: http://agilemanifesto.org/.

Becker, H. S. (1996) *The Epistemology of Qualitative Research*. Accessed on 29/5/2002 at: http://www.soc.ucsb.edu/faculty/hbecker/qa.html.

Beesley, M. E. (ed.) (1994) *Regulating Utilities: The Way Forward*. The Institute of Economic Affairs.

Beesley, M. E. (1997) *Privatization, Regulation and Deregulation*. 2nd Edition, Routledge.

Beggs, A. and Klemperer, P. D. (1992) Multiperiod Competition with Switching Costs. *Econometrica*, vol. 60, pp. 651-666.

Beizer, B. (2001) *Software Is Different*. Accessed on 08/08/2002 at: http://www.soft.com/News/QTN-Online/qtnapr01.html.

Benbasat, I., Goldstein, D. K. and Mead, M. (1987) The Case Research Strategy in Studies of Information Systems. *MIS Quarterly*, vol. 11, no. 3, pp. 369-386.

Benbasat, I. And Zmud, R.W. (1999) Empirical Research in Information Systems: The Practice of Relevance. *MIS Quarterly*, vol. 23, no. 1, pp. 3-16.

Bennett, R. (1991) How is management research carried out? In N. C. Smith and P. Dainty (eds.) *The Management Research Handbook*, Routledge, pp. 85-103.

Bierman, H. S. and Fernandez, L. (1998) *Game Theory with Economic Applications*. Addison Wesley.

Birkinshaw, J. (2001) Strategies for Managing Internal Competition. *California Management Review*, vol. 44, no. 1 (Fall), pp. 21-38.

Blanchette, I. and Dunbar, K. (2000) *How Analogies are Generated: The Roles of Structural and Superficial Similarity*. Accessed on 10/07/2003 at: http://www.dartmouth.edu/~kndunbar/Blan%26DunMC2000.pdf.

Blin, M. J. and Tsoukias, A. (2001) Multicriteria methodology contribution to software quality evaluations. *Software Quality Journal*, vol. 9, no. 2, pp. 113-132.

Bocij, P. Chaffey, D. Greasley, A. and Hickie, S. (1999) *Business Information Technology, Technology, Development and Management*. Pitman Publishing.

Boehm, B. (1976) Software Engineering. *IEEE Transactions on Computers*, vol. C-25, no. 12, pp. 1226-41. Also in E. Yourdon (ed., 1979), *Classics in Software Engineering*. Yourdon Press, pp. 325-361.

Boehm, B. (2000a) *Unifying software engineering and system engineering*. Accessed on 14/12/2002 at: http://sunset.usc.edu/publications/TECHRPTS/2000/usccse2000-506/usccse2000-506.pdf.

Boehm, B. (2000b) Requirements that Handle IKIWISI, COTS, and Rapid Change. *Computer*, vol. 33, no. 7, pp. 99–102.

Booth, W. C., Colomb, G. G. and Williams, J. M. (1995) *The Craft of Research*. The University of Chicago Press.

Bott, F., Coleman, A., Eaton, J. and Rowland, D. (2001) *Professional Issues in Software Engineering*. 3rd Edition. Taylor & Francis.

Boudreaux, K. (1990) *Finance: A Distance Learning Programme*. Pitman Publishing.

Bradbury, D. (2002) *Better the devil you know*. Accessed on 17/10/2003 at: http://www.computerweekly.com/Article115904.htm.

Brewer, J. and Hunter A. (1989) *Multimethod Research: A Synthesis of Styles*. Sage Publications.

Britton, C. and Doake, J. (1996) *Software System Development: A gentle Introduction*. 2nd Edition. McGraw-Hill.

Bronzite, M. (2000) *System Development, A Strategic Framework.* Springer-Verlag.

Brooks, F. P. (1975) *The Mythical Man-Month.* Addison Wesley.

Brooks, F. P. (1995) *The Mythical Man-Month.* 20th Anniversary Edition, Addison Wesley.

Brown, A. D., Boyett, I. and Robinson, P. (1994) The Dynamics of Partnership Sourcing. *Leadership & Organization Development Journal*, vol. 15 no. 7, pp. 15-18.

Brownsword, L., Carney, D. and Oberndorf, T. (1998) *The Opportunities and Complexities of Applying Commercial-Off-the-Shelf Components.* Accessed on 08/07/2003 at: http://www.stsc.hill.af.mil/crosstalk/1998/04/applying.asp.

Buchanan, M., Iyer, R. and Karl, C. A. (1999) *The Case Study in Business Research.* Accessed on 05/05/2002: http://www.bechervaise.com/DBAR3.htm.

Burke, G. and Peppard, J. (eds.) (1995) *Examining Business Process Re-Engineering: Current Perspectives and Research Directions.* Kogan Page.

Buxbaum, P. (2001) *See You in Court.* Accessed 09/01/2002 at: http://www.computerworld.com/cwi.

Cabinet Office (2000) *Successful IT: Modernising Government in Action.* Accessed on 21/01/2003 at: http://www.ogc.gov.uk/index.asp?docid=2632.

Callahan, J. and Moretton, B. (2001) Reducing software product development time. *International Journal of Project Management*, vol. 19, no. 1, pp. 59-70.

Carmel, E. and Sawyer, S. (1998) Packaged software development teams: what makes them different? *Information Technology & People*, vol. 11, no. 1, pp. 7-19.

CCTA (1997) *Project Management Industry Initiatives.* Central Computer and Telecommunications Agency. Accessed on 28/05/2003 at: http://www.ogc.gov.uk/PRINCE.

CCTA (1998) *Managing Successful Projects with PRINCE 2.* Central Computer and Telecommunications Agency. The Stationery Office.

Chan, E. H. W. and Chan, A. P. C. (2000) *Design-Build Contracts In Hong Kong- Some Legal Concerns.* (Abstract). Accessed on 25/04/2003 at: http://www.ing.puc.cl/~icccon/abstracts/PDF/Track3/T3-P7.pdf.

Chapman, C. (2000) Project Risk Management: The Required Transformations to Become Project Uncertainty Management. In *Project Management Research at the Turn of the Millennium, Proceedings of PMI Research Conference 2000, 21-24 June 2000, Paris, France.* Project Management Institute, pp. 241-245.

Checkland, P. (1981) *Systems Thinking, Systems Practice.* John Wiley.

Clark, K. B. and Fujimoto, T. (1991) *Product Development Performance: Strategy, Organization and Management in the World Auto Industry.* Harvard Business School Press.

Cockburn, A. and Williams, L. (2000) *The Costs and Benefits of Pair Programming.* Accessed on 16/07/2003 at: http://collaboration.csc.ncsu.edu/laurie/Papers/XPSardinia.PDF.

Cohen, L. (1995) *Quality Function Deployment: How to Make QFD Work for You.* Addison Wesley Longman.

Cole, R. E. (2001) From Continuous Improvement to Continuous Innovation. *Quality Management Journal*, vol. 8, no. 4. Accessed on 17/10/2003 at: http://www.asq.org/pub/qmj/past/vol8_issue4/cole.html.

Collins, T. (1998) *Crash: Learning from the World's Worst Computer Disaster.* Simon & Schuster.

Computing Staff (2003) *Claims over troubled Libra project denied. Computing,* 12/02/2003. Accessed on 07/05/2003 at: http://www.computing.co.uk/News/1138719.

Cooper, K. G., Lyneis, J. M. and Bryant, B.J. (2002) Learning to learn, from past to future. *International Journal of Project Management*, vol. 20, no. 3, pp. 213-219.

Corder, C. (1985) *Ending the Computer Conspiracy: The Thinking Person's Guide to Successful Systems.* McGraw-Hill.

Correia, C., Flynn, D. K., Uliana, E. and Wormald, M. (1989) *Financial Management.* 2nd Edition. Juta & Co.

Cousins, P. D. (2002) A conceptual model for managing long-term inter-organisational Relationships. *European Journal of Purchasing & Supply Management.* vol. 8, pp. 71–82.

Crawford, L. (2000) Profiling the Competent Project Manager. *In Project Management Research at the Turn of the Millennium, Proceedings of PMI Research Conference 2000, 21-24 June 2000, Paris, France.* Project Management Institute, pp. 3-15.

Crawford, I. (2003) *Risk Management Write-up.* Accessed on 28/08/2003 at: http://www.pmi.org.uk/newslet/apr2003/riskman.htm.

Currie, W. (1997) Computerising the Stock Exchange: a comparison of two information systems. *New Technology, Work and Employment*, vol. 12, no. 2, pp. 75-83.

Curtis, B., Krasner, H. and Iscoe, N. (1988) A Field Study of the Software Design Process for Large Systems. *Communications of the ACM.* vol. 31, no. 11, pp. 1268-1287.

D'Adderio, L. (2000) *The Diffusion Of Integrated Software Solutions: Trends And Challenges.* Accessed on 08/07/2003 at: http://www.cespri.it/ricerca/dadderio.PDF.

Davenport, T. H. and Markus, M. L. (1999) Rigor vs. Relevance Revisited: Response to Benbasat and Zmud. *MIS Quarterly*, vol. 23, no. 1, pp. 19-23.

Davidson, A. L. (2001) *Grounded Theory.* Accessed on 30/05/2002 at: http://az.essortment.com/groundedtheory_rmnf.htm.

Davis, C. (undated) *Track Defects Against Quality Targets.* Accessed on 08/09/2003 at: http://www.2asc.com/ascweb/products%20and%20services/risk/Best%20Practices/content/5_defect/dftfrm.htm .

Davis, G. B. (2000) Information Systems Conceptual Foundations: Looking Backward And Forward. In R. Baskerville, J. Stage and J. I. DeGross, (eds.), *Organizational and Social Perspectives on Information Technology, Part 2: Transforming the Fundamentals*. Kluwer, pp. 61-82. Accessed on 26/05/2003 at: www.terry.uga.edu/~ekarah/davis.pdf.

Davison, R. M. (1998) *An Action Research Perspective of Group Support Systems: How to Improve Meetings in Hong Kong*. PhD Thesis of City University of Hong Kong. Accessed on 12/5/2002 at: http://www.is.cityu.edu.hk/People/AcademicStaff/rd/rd-phd.htm.

Day, M. (1998) *The Process of Research*. Accessed on 12/05/2002 at: http://www.keele.ac.uk/depts/mn/pgrad/process.html.

DeMarco, T. (1979) *Structured Analysis and Systems Specifications*. Yourdon Press.

DeMarco, T. and Lister, T. (1999) *Peopleware: Productive Projects and Teams*. 2nd Edition. Dorset House Publishing.

Dench, S., Iphofen, R. and Huws, U. (2003) *Draft Guidelines for Conducting Ethical Socio-Economic Research*. Accessed on 28/08/2003 at: http://www.respectproject.org/ethics/ethics_report_0603.pdf.

Denzin, N. K. and Lincoln, Y. S. (1998) *The Landscape of Qualitative Research, Theories and Issues*. Sage Publications.

Denzin, N. K. and Lincoln, Y. S. (2000) The Discipline and Practice of Qualitative Research. In N. K. Denzin and Y. S. Lincoln (eds.), *Handbook of Qualitative Research*. 2nd Edition. Sage Publications, pp. 1-28.

Dhamija, R. and Perrig, A. (2000) *Déjà Vu: A User Study Using Images for Authentication*. Accessed on 16/07/2003 at: http://citeseer.nj.nec.com/326534.html.

Dick, B. (2000) *Grounded Theory: A Thumbnail Sketch*. Accessed on 29/05/2002 at http://www.scu.edu.au/schools/gcm/ar/arp/grounded.html.

Dooijes, E. H. (1999) *Trends In Computer Technology*. Accessed on 27/10/2003 at: http://www.science.uva.nl/faculteit/museum/technotrends.html.

Drummond, H. (1999) Are we any closer to the end? Escalation and the case of Taurus. *International Journal of Project Management*, February 1999, vol. 17, no. 1, pp. 11-16.

DSDM (undated-a) *The History of DSDM*. Accessed on 15/10/2002 at: http://www.dsdm.com/en/about/history.asp.

DSDM (undated-b) *Overview: Why is DSDM different*. Accessed on 07/07/2003 at: http://www.dsdm.com/en/about/overview.asp.

Dunbar, K. (1995). *How scientists really reason: Scientific reasoning in real-world laboratories*. Accessed on 10/07/2003 at: http://www.dartmouth.edu/~kndunbar/DunbarStern.pdf.

Dyer, J. and Ouchi, W. (1993) Japanese-style partnerships: Giving companies a competitive edge. *Sloan Management Review*, vol. 35, pp. 51–63.

Dyer, W. G. Jr. and Wilkins, A. L. (1991) Better stories, not better constructs, to generate better theory: a rejoinder to Eisenhardt. *Academy of Management Review*, vol. 16, no. 3, pp. 613-619.

EA (2003) *Who owns whom in the UK electricity industry.* Electricity Association. Accessed on 10/07/2003 at: http://www.electricity.org.uk/media/documents/pdf/who_owns_whom.pdf.

Eden, C. and Huxham, C. (1996) Action research for the study of organizations. In S. Clegg, C. Hardy and W. Nord (eds.) *Handbook of organization studies*, Sage, pp. 526–542.

Eischen, K. (2002) *Software Development: A View from the Outside.* Accessed on 14/07/2003 at: http://www2.ucsc.edu/cgirs/publications/wp/wp2002-3.pdf.

Eisenhardt, K. M. (1989a) Agency Theory: An Assessment and Review. *Academy of Management Review*, vol. 14, no. 1, pp. 57-74.

Eisenhardt, K.M. (1989b) Building theories from case study research. *Academy of Management Review*, vol. 14, no. 4, pp. 532-550.

Elgood, C. (1984) *Handbook of Management Games.* 3rd Edition. Gower.

Ellram, L. M. (1991) A Managerial Guideline for the Development and Implementation of Purchasing Partnerships. *The Journal of Supply Chain Management.* vol. 27, no. 3, pp. 2-8.

Farrell, J. and Klemperer, P. (2001) *Coordination and Lock-In: Competition with switching Costs and Network Effects*. (2001 draft). Accessed on 21/08/2002 at: http://emlab.berkeley.edu/users/farrell/ftp/lockin.pdf.

Farrell, J. and Shapiro, C. (1989) Optimal Contracts with Lock-In. *The American Economic Review*, vol. 79, no. 1, pp. 51-68.

Ferguson, P. R., and Ferguson, G. J. (1994) *Industrial Economics: Issues and Perspectives.* 2nd Edition. The MacMillan Press.

Ferris, N. (1999) *ERP: Sizzling or Stumbling?* Accessed on 09/10/2002 at: http://www.govexec.com/news/index.cfm?mode=report&articleid=18080&printerfriendlyVers=1&.

Festa, P. (2001) *Software to blame for security problems.* Accessed on 27/10/2003 at: http://news.zdnet.co.uk/story/0,,t269-s2100224,00.html.

Feyerabend, Paul (1995) *Killing Time: The Autobiography of Paul Feyerabend.* The University of Chicago Press.

Field, T. (1997) *When Bad Things Happen to Good Projects.* Accessed on 16/01/2002 at: http://www.cio.com/archive/101597/bad_content.html?printversion=yes.

Field, R. (2001) John Dewey (1859-1952). *The Internet Encyclopedia of Philosophy.* Accessed on 12/5/2002 at: http://www.utm.edu/research/iep/d/dewey.htm.

Finkelstein, A. (2001) CAPSA And Its Implementation Report To The Audit Committee And The Board Of Scrutiny: Part A. *Cambridge University Reporter*, No5861, 2 November 2001. Accessed on 17/07/2003 at: http://www.admin.cam.ac.uk/reporter/2001-02/weekly/5861/.

Fitzgerald, B (2000) Systems Development Methodologies: The Problem of Tenses. *Information Technology & People*, vol. 13, no. 3, pp. 174-185.

Flick, U. (1998) *An Introduction to Qualitative Research.* Sage Publications.

Flowers, S. (1995) *Software Failure: Management Failure.* John Wiley & Sons.

Friedman, A. L. (1989) *Computer Systems Development: History, Organization and Implementation.* John Wiley & Sons.

Gage, D. (2002) *How Carreker Manages PeopleSoft.* Accessed on 15/12/2002 at: http://www.baselinemag.com/article2/0,3959,36483,00.asp.

Galliers, R. D., (1985) In search of a paradigm for information systems research. In Mumford, E. et al. (eds.), *Research Methods in Information Systems*, Elsevier, pp. 281-297.

Galliers, R. D. (1995) IT and Organizational Change: Where does BPR fit in? In G. Burke and J. Preppard (eds.), *Examining Business Process Re-Engineering: Current Perspectives and Research Directions*. Kogan Page, pp. 117-134.

Gane, C. and Sarson, T. (1979) *Structured Systems Analysis: Tools and Techniques.* Prentice-Hall.

Ganssle, J. (2002) *Measuring Bugs.* Accessed on 08/09/2003 at: http://www.embedded.com/story/OEG20020625S0039.

GAO (1992a) *Tax Systems Modernization, IRS Award To Mitre Corporation Violated The Competition In Contracting Act.* The United States General Accounting Office, GAO/IMTEC-92-28.

GAO (1992b) *Patriot Missile Defense: Software Problem Led to System Failure at Dhahran, Saudi Arabia.* The United States General Accounting Office, GAOAMTEC-92-26.

GAO (1994) *New Denver Airport: Impact Of The Delayed Baggage System.* The United States General Accounting Office, GAO/RCED-95-35BR.

GAO (1997) *Battlefield Automation: Software Problems Hinder Development of the Army's Maneuver Control System.* The United States General Accounting Office, GAO/NSIAD-98-15.

GAO (2002) *Information Technology: Inconsistent Software Acquisition Processes at the Defense Logistics Agency Increase Project Risks.* The United States General Accounting Office, GAO-02-9.

Gardiner, P.D. and Stewart, K. (2000) Revisiting the golden triangle of cost, time and quality: the role of NPV in project control, success and failure. *International Journal of Project Management*, vol. 18, pp. 251-256.

Gardner, R. (1995) *Games for Business and Economics.* John Wiley & Sons.

Gattiker, T. F. and Goodhue, D. L. (2002) Software-driven changes to business processes: an empirical study of impacts of Enterprise Resource Planning (ERP) systems at the local level. *International Journal of Production Research,* vol. 40, no. 18, pp. 4799-4814.

GenerExe (2002) *A short History of Software Development*. Accessed on 16/07/2003 at: http://www.generexe.com/softhist.pdf.

George, K. D., Joll, C. and Lynk, E. L. (1991) *Industrial Organisation: Competition, Growth and Structural Change*. 4th Edition. Routledge.

Gibbons, R. (1992) *A Primer in Game Theory*. Harvester Wheatsheaf.

Gibbs, W. (1994) Software's Chronic Crisis. *Scientific American*, vol. 271, no. 3, pp. 86-95.

Gilb, T. (1997) *Requirements-Driven Management: A Planning Language*. Accessed on 14/08/2002 at: http://www.stsc.hill.af.mil/SWTesting/gilb.asp.

Girard, K. and Farmer, M. A. (1999) *Business software firms sued over implementation*. Accessed on 17/01/2003 at: http://news.com.com/2100-1001-232404.html.

Giulietti, M. and Price, C. W. (2000) *Incentive Regulation and Efficient Pricing: Empirical Evidence*. Accessed on 10/07/2003 at: http://users.wbs.warwick.ac.uk/cmur/publications/research_paper_002.pdf.

Glaseman, S. (1982) *Comparative Studies in Software Acquisition*. LexingtonBooks.

Glaser, B. G. and Strauss, A. L. (1967) *The Discovery of Grounded Theory: Strategies For Qualitative Research*. Aldine.

Glass, R. L. (1997) Revisiting the Industry/Academe Communication Chasm. *Communications of the ACM*, vol. 40, no. 6, pp. 11-13.

Glass, R. L. (1998) *Software Runaways: Lessons Learned from Massive Software Project Failures*. Prentice Hall.

Glass, R. L. (1999) *Computing Calamities: Lessons Learned from Products, Projects, and Companies that Failed*. Prentice Hall.

Gorb, P. (1990) *Design Management*. Architecture Design and Technology Press.

Guba, E. G. and Lincoln, Y. S. (1998) "Competing paradigms in qualitative research", in N. K. Denzin & Y. S. Lincoln (Eds.), *The Landscape of Qualitative Research: Theories and Issues*, Sage, pp. 195-220.

Gulliver, F. R. (1987) Post-project appraisals pay. *Harvard Business Review*, March–April, pp. 128–131.

Haig, B. D. (1995) Grounded Theory as Scientific Method. Access on 11/04/2002 at: http://www.ed.uiuc.edu/EPS/PES-yearbook/95_docs/haig.html.

Hammersley, M. (1989) The dilemma of qualitative methods – Herbert Blumer and the Chicago Tradition. Routledge.

Harman, G. H. (1965) The Inference to the Best Explanation. *The Philosophical Review*, vol. 74, no. 1, pp. 88-95.

Hart, O. and Moore, J. (1999) Foundations of Incomplete Contracts. *Review of Economic Studies*, vol. 66, pp. 115-138.

Hayes, M. (2002) *Quality First.* Accessed on 19/11/2002 at: http://www.informationweek.com/story/IWK20020517S0010.

Helm, D. (1994) Regulating the transition to the competitive electricity market. In M. E. Beesley (ed.), *Regulating Utilities: The Way Forward.* The Institute of Economic Affairs, pp. 89-114.

Helpsoft (undated) *Why DSDM is different.* Accessed on 06/09/2003 at: http://www.helpsoft.net/corporate/ebusiness/why_helpsoft_is_different.htm.

Henning, W. (2001) *Pentium 4 2GHz Review.* Accessed on 09/01/2003 at: http://www.cpureview.com/rev_p4-2GHz_a.html.

Hertz, D. B. and Thomas, H. (1983) *Risk Analysis and its Applications.* John Wiley & Sons.

Highsmith, J. (2000) *Adaptive Software Development.* Dorset House, New York.

Hobsons (2002) *Graduate Career Directory 2002.* Hobsons.

Hollanders, H. and Meijers, H. (2001) *Quality-Adjusted Prices And Software Investments: The Use Of Hedonic Price Indexes.* Accessed on 17/11/2002 at: http://www.researchineurope.org/newkind/documents/hugo1.doc.

Holyoak, K. J. and Thagard, P. (1997) The analogical mind. *American Psychologist*, vol. 52, no. 1, pp. 35-44. Accessed on 10/07/2003 at: http://cogprints.ecs.soton.ac.uk/archive/00000658/00/Analog_mind.html.

Hopper, M. D. (1990) Rattling SABRE – new ways to compete on information. *Harvard Business Review*, May-June, pp. 118-25.

Howard, A. (2002) Rapid Application Development: Rough and Dirty or Value-for-Money Engineering? *Communications of the ACM*, vol. 45, no. 10, pp. 27-29.

Howard, P. (2003) *Reference Data Management.* Accessed on 27/06/2003 at: http://www.it-director.com/article_pf.php?articleid=3592.

Howe, D. R. (1983) *Data Analysis for Data Base Design.* Arnold.

Humphrey, W. S. (1989) *Managing The Software Process.* Addison-Wesley.

Humphrey, W. S. (1998) *Why Don't They Practice What We Preach?* Accessed as of 6/11/2002 at: http://www.sei.cmu.edu/publications/articles/practice-preach/practice-preach.html.

Hutchinson Reference (1991) *The Hutchinson Softback Encyclopedia.* Random Century.

Hynes, M., Kirby, S. N. and Sloan, J. (2000) *A Casebook of Alternative Governance Structures and Organizational Forms.* Accessed on 29/08/2003 at: http://www.rand.org/publications/MR/MR1103.

Iivari, J., Hirschheim, R. and Klein, H. K. (2001) A Dynamic Framework for Classifying Information Systems Development Methodologies and Approaches. *Journal of Management Information Systems*, vol. 17, no. 3 (Winter), pp. 179-218.

Inacio, C. (1998) *Software Fault Tolerance.* Accessed on 16/07/2003 at: http://www.ece.cmu.edu/~koopman/des_s99/sw_fault_tolerance/.

INCOSE (2000) *Systems Engineering Handbook: A "How To" Guide For All Engineers.* International Council of Systems Engineering.

Jarzombek, J. (1998) *The Double-Edged COTS IT Sword.* Accessed on 08/07/2003 at: http://www.stsc.hill.af.mil/crosstalk/1998/04/applying.asp.

Jayaratna, N. (1994) *Understanding and Evaluating Methodologies.* McGraw-Hill.

Jayaratna, N. and Holt, P. (1996) Selection criteria for methodologies. *International Journal of Information Management*, vol. 16, no. 1, pp. 75-76.

JCP (2002) *The Java Community Process, Sun Microsystems.* Accessed on 28/11/2002 at: http://jcp.org/en/home/index.

Jensen, R. W. (2003) *A Pair Programming Experience.* Accessed on 16/07/2003 at: http://www.stsc.hill.af.mil/crosstalk/2003/03/jensen.html.

Jobber, D. (1991) Choosing a survey method in management research. In N. C. Smith and P. Dainty (eds.) *The Management Research Handbook*, Routledge, pp. 174-180.

Johnson, J. (1999) *Turning CHAOS into SUCCESS.* Accessed on 22/11/2002 on: http://www.softwaremag.com/L.cfm?Doc=archive/1999dec/Success.html.

Johnson, J. (2000a) *Top 10 Pocket-Pickers.* Accessed on 22/11/2002 at: http://www.softwaremag.com/L.cfm?Doc=archive/2000oct/Top10PocketPickers.html.

Johnson, J. (2000b) *How to Succeed with Performance-based Contracts.* Accessed on 22/11/2002 at: http://www.softwaremag.com/L.cfm?Doc=archive/2000oct/Performance.html.

Johnson, J. A. (2000) *Abductive Inference and the Problem of Explanation in Social Science.* Accessed on 12/09/2002 at: http://polisci.wisc.edu/~jajohnson/research/mpsa2000.pdf.

Jones, C. (1996) *Patterns Of Software Systems Failure And Success.* Thomson.

Jones, M. R. (1995) The Contradictions of Business Process Re-Engineering. In G. Burke and J. Preppard (eds.), *Examining Business Process Re-Engineering: Current Perspectives and Research Directions.* Kogan Page, pp. 43-59.

Josephson, J. R (1996) *Inductive generalisations are abductions.* Accessed on 24/5/2002 at: http://www.cs.bris.ac.uk/~flach/ECAI96/papers.html.

Josephson, J. R. (2001) *On The Proof Dynamics Of Inference To The Best Explanation.* Accessed on 24/5/2002 at: http://www.cardozo.yu.edu/cardlrev/v22n5-6/josephson.pdf.

Josittus, N. (2001) *Designing a 3-Tier-Architecture.* Accessed on 29/11/2002 at: http://www.cs.wustl.edu/~mk1/ThreeTierPatterns/submissions/NicolaiJosuttis.pdf.

Kador, J. (2001) *Leveraging process improvement.* Accessed on 04/01/2002 at: http://www.findarticles.com/cf_dls/m0IFW/12_23/71847965/p1/article.jhtml.

Kaner, C. and Pels, D. (1998) *Bad Software: What To Do When Software Fails.* John Wiley & Sons.

Karlsson, C., Nellore, R. and Söderquist, K. (1998) Black Box Engineering: Redefining the Role of Product Specifications. *Journal of Product Innovation Management*, vol.15, no.6, pp. 534-549.

Katz, D. M. (2001) *Of Men and Mice: An ERP Case Study.* Accessed on 15/01/2003 at: http://www.cfo.com/article/1,5309,2348|0|||,00.html.

Kay, J. (1994) Regulating Networks. In M. E. Beesley (ed.), *Regulating Utilities: The Way Forward.* The Institute of Economic Affairs, pp. 77-88.

Keil, M. (2000) Cutting Your Losses: Extricating Your Organization When a Big Project Goes Awry. *Sloan Management Review*, vol. 41, Spring, pp. 55-68.

Keil, M., Tiwana, A. and Bush, A. (2002) Reconciling user and project manager perceptions of IT project risk: a Delphi Study. *Information Systems Journal*, vol. 12, pp. 103–119.

Kelle, U. (2001) *Sociological Explanations between Micro and Macro and the Integration of Qualitative and Quantitative Methods.* Accessed on 01/05/2002 at: http://qualitative-research.net/fqs-texte/1-01/1-01kelle-e.htm.

Kenwood, C. A. (2001) *A Business Case Study of Open Source Software.* Accessed on 27/02/2003 at: www.mitre.org/support/papers/tech_papers_01/kenwood_software/kenwood_software.pdf.

Kerzner, H. (1998) *Project Management: A SystemsApproach to Planning, Scheduling, and Controlling.* 6th Edition. John Wiley & Sons.

Keser, C. and Willinger, M. (2000) Principals' principles when agents' actions are hidden. *International Journal of Industrial Organization.* vol. 18, pp. 163-185.

Kharbanda, O. P. and Pinto, J. K. (1996) *What Made Gertie Gallop? Lessons from Project Failures.* Van Nostrand Reinhold.

Klemperer, P. (1987) The competitiveness of markets with switching costs. *Rand Journal of Economics*, vol. 18, no. 1, pp. 138-150.

Klemperer, P. (1989) Price Wars Caused by Switching Costs. *Review of Economic Studies*, vol. 56, pp. 405-420.

Klemperer, P. (1995) Competition when Consumers have Switching Costs: An Overview with Applications to Industrial Organization, Macroeconomics, and International Trade Review of Economic Studies. *Review of Economic Studies*, vol. 62, pp. 515-539.

Kock, N. (1997) Myths in Organisational Action Research: Reflections on a Study of Computer-Supported Process Redesign Groups. *Organizations & Society*, vol. 4, no. 9, pp. 65-91.

Kock, N. and Lau, F. (2001), Information Systems Action Research: Serving Two Demanding Masters. *Information Technology & People*, Special Issue on Action Research in Information Systems, vol. 14, no. 1, pp. 6-11.

Kohl, R. and Oberndorf, P. (2000) *The Software Spectrum: A Shopper's Guide.* Accessed on 15/02/2002 at: http://wwwsel.iit.nrc.ca/projects/cots/icse2000wkshp/Papers/30.pdf.

Kontio, J. (1996) *A Case Study in Applying a Systematic Method for COTS Selection.* Accessed on 09/01/2003 at: http://www.soberit.hut.fi/~jkontio/icse18_otsocase.pdf.

Koskela, L. and Howell, G. (2002a) *The underlying theory of project management is obsolete.* Accessed on 27/02/2004 at: http://www.leanconstruction.org/pdf/ObsoleteTheory.pdf.

Koskela, L. and Howell, G. (2002b) *The Theory Of Project Management: Explanation To Novel Methods*. Accessed on 10/03/2004 at: http://www.cpgec.ufrgs.br/norie/iglc10/papers/47-Koskela&Howell.pdf.

KPMG (1997) *What Went Wrong? Unsuccessful Information Technology Projects.* KPMG Canada.

Kruchten, P. (2000) *The Rational Unified Process: An Introduction.* 2nd Edition. Addison-Wesley.

Kuhn, T. S. (1970) *The Structure of Scientific Revolutions.* 2nd Edition. The University of Chicago Press.

Kumar, A. (2001) *Global IT spend to touch $1.746 trillion by 2005.* Accessed on 29/10/2003 at: http://www.rediff.com/money/2001/may/31it.htm.

Kuwabara, K. (2000) *Linux: A Bazaar at the Edge of Chaos.* Accessed on 06/11/2002 at: http://firstmonday.org/issues/issue5_3/kuwabara/index.html.

Kwakye, A. A. (1997) *Construction Project Administration in Practice.* Longman.

Lacity, M. C. and Hirchheim, R. (1995) *Information Systems Outsourcing: Myths, Metaphors and Realities.* John Wiley & Sons.

Lacity, M. C. and Willcocks, L. P. (2000) *Relationships in IT Outsourcing: A Stakeholder Perspective.* Accessed on 23/01/2002 at: http://www.templeton.ox.ac.uk/pdf/researchpapers/restricted/00-09.pdf.

Lakatos, I. (1978) *The Methodology of Scientific Research Programmes.* Cambridge University Press.

Lamming, R. and Cox, A. (eds.) (1995) *Strategic Procurement Management In The 1990s: Concepts And Cases.* Earlsgate Press.

Lantz, K. E. (1987) *The Prototyping Methodology.* Prentice-Hall.

Larman, C. and Basili, V. R. (2003) Iterative and Incremental Development: A Brief History. *IEEE Computer*, June, pp. 47-56.

Layman, B. (2001) *An Interview with Jerry Weinberg*. Accessed on 19/07/2002 at: http://www.stickyminds.com/se/S3539.asp.

Lederer, A. L. and Prasad, J. (2000) Software Management and Cost Estimating Error. *The Journal of Systems and Software*, vol. 50, pp. 33-42.

Lerner, J. and Tirole, J. (2000) *The Simple Economics of Open Source*. Accessed on 27/02/2003 at: http://www.people.hbs.edu/jlerner/simple.pdf.

Levy, L. S. (1987) *Taming the Tiger: Software Engineering and Software Economics*. Springer-Verlag.

Levy, S. M. (2002) *Project Management in Construction*. 4th Edition. McGraw-Hill.

Lewis, P., Hyle, P., Parrington, M. et al. (2000) *Lessons Learned in Developing Commercial Off-The-Shelf (COTS) Intensive Software Systems*. Accessed on 25/11/2002 at: http://www.cebase.org/www/researchActivities/COTS/LessonsLearned.pdf.

Lichtenstein, Y. (2002) *Puzzles in Software Development Contracting*. Accessed on 05/07/2003 at: http://mis.ucd.ie/staff/Ylichtenstein.

Lieberman, B. A. (2002) *Gambling with Success: Software Risk Management*. Accessed on 01/09/2003 at: http://www.therationaledge.com/content/jul_02/m_gamblingWithSuccess_bl.jsp.

Lincoln, Y. S. and Guba, E. G. (1985) *Naturalistic Inquiry*, Sage, Beverly Hills.

Lincoln, Y. S. and Guba, E. G. (2000) Paradigmatic controversies, contradictions, and emerging confluences. In N.K. Denzin and Y.S. Lincoln (eds.), *Handbook of Qualitative Research*, 2nd Edition. Sage Publications, pp. 163-188.

Lindberg, P. (2003) *Mary & Tom Poppendieck: Lean Development (part 2)*. Accessed on 16/07/2003 at: http://tesugen.com/irrational/archives/maryAndTomPoppendieckLeanDevelopmentPart2.html

Linuxcare (2000) *Demystifying Open Source: How Open Source Software Development Works*. Accessed on 27/02/2003 at: http://www.linuxcare.com/about-us/collateral/index.epl.

Lipton, P. (1991) Inference To The Best Explanation. Routledge.

Livesey, F. (1993) *Dictionary of Economics*. Longman.

Locke, K. (2000) *Grounded Theory in Management Research*. Sage Publications.

Lothian, N. and Small, J. (1991) *Accounting: A Distance Learning Programme*. Pitman Publishing.

Loverica, G. (1994) *New 64-bit graphics accelerator cards break Windows performance barriers*. Accessed on 09/01/2003 at: http://www.byte.com/art/9407/sec10/art2.htm.

Lowe, A. (1996) *Grounded Theory*. A videocassette made by Department of Marketing. University of Stirling.

Lowe, T. J. and Wendall, R. E. (2000) Managing Risks in Projects With Decision Technologies. In *Project Management Research at the Turn of the Millennium, Proceedings of PMI Research Conference 2000, 21-24 June 2000, Paris, France.* Project Management Institute, pp. 61-72.

Lumsden, K. G. (1995) *Economics: a distance learning study programme.* Pitman Publishing.

Lyytinen K. and Robey D. (1999) Learning Failure in Information System Development, *Information Systems Journal*, vol. 9, no. 2, pp. 85-101.

MacBeth, D. K. and Ferguson, N. (1994) *Partnership Sourcing.* Pitman Publishing.

Machiavelli, N (1999) *The Prince.* (First published in 1516). Penguin Classics.

Macromedia (undated) *Macromedia Director Support Center Basics: Design Phase.* Accessed on 08/09/2003 at: http://www.macromedia.com/support/director/how/expert/manage/managemm02.html.

Maddison, R.N. (1983) *Information System Methodologies.* Wiley Heydon.

Mailath, G. J. and Samuelson, L. (2001) Who Wants a Good Reputation? *Review of Economic Studies*, vol. 68, pp. 415-441.

Malgrati, A. and Damiani, M. (2002) Rethinking the new project management framework: new epistemology, new insights. *Proceedings of PMI (Project Management Institute) Research Conference*, Seattle 2002, pp. 371-380.

Marciniak, J. J. and Reifer, D. J. (1990) *Software Acquisition Management: Managing the Acquisition of Custom Software Systems.* John Wiley & Sons.

Marshall, M. K. (1992) *Design to confuse.* Accessed on 03/03/2004 at: http://www.trett.com/digest/doit.asp?iss=6&art=2 and http://www.trett.com/digest/doit.asp?iss=7&art=3.

Martin. R. C. (1999) *Iterative and Incremental Development.* Accessed on 30/07/2003 at: www.objectmentor.com.

Maskin, E. (2002) Incomplete Contracts: On indescribable contingencies and incomplete contracts. *European Economic Review*, vol. 46, pp. 725–733.

Maude, T. and Willis, G. (1991) *Rapid Prototyping: The Management of Software Risk.* Pitman Publishing.

Maxman, R. M. (2003) *Software Development and Linearity (Or, why some project management methodologies don't work), Part 1 and Part 2.* Accessed on 23/02/2004 at: http://www.maxwideman.com/papers/linearity/linearity1.pdf http://www.maxwideman.com/papers/linearity/linearity2.pdf.

May, L. J. (1998) *Major Causes of Software Project Failures.* Accessed on 09/01/2002 at: http://www.stsc.hill.af.mil/crosstalk/1998/07/causes.pdf.

Mazur, G. H. (1993) *QFD for Services Industries: from voice of customers to task deployment.* Accessed on 29/10/2003 at: http://www.mazur.net/works/sysctaskqfd.pdf.

McBreen, P. (2000) *Uncertainty in Software Development.* Accessed on 21/08/2003 at: http://www.mcbreen.ab.ca/papers/Uncertain.html.

McBreen, P. (2002a) *Software Development: Dismantling the Waterfall.* Accessed on 22/08/2003 at: http://www.informit.com/content/index.asp?product_id=%7B92C94D40-9CA8-45E0-A5CF-DE466A8FD5AD%7D.

McBreen, P. (2002b) *Exposing the Fallacy of "Good Enough" Software.* Accessed on 17/06/2003 at: http://www.awprofessional.com/authors/author.asp?authorid={7B7D1926-0DED-4713-A53B-549AF36312DE.

McConnell, S. (1993) *Code Complete: A Practical Handbook of Software Construction.* Microsoft Press International.

McConnell, S. (1996) *Classic Mistakes.* Accessed on 29/10/2003 at: http://www.stevemcconnell.com/bp05.htm.

McConnell, S. (1997) *Managing Outsourced Projects.* Accessed on 28/08/2003 at: http://www.stevemcconnell.com/articles/art07.htm.

McGloin, E. and Grant, C. (1998) Supporting partnership sourcing in Northern Ireland through advanced technology. *Technovation*, vol. 18, no. 2, pp. 91-99.

McManus, J. (1997) *If you want to succeed in software development...try Rapid Applications Development (RAD).* Accessed on 16/07/2003 at: http://www.bcs.org.uk/publicat/ebull/feb97/rap.htm.

McMillan, J. (1990) Managing Suppliers: Incentive Systems in Japanese and United States Industry. *California Management Review*, vol. 32, pp. 38-55.

McPartlin, J. P. (1992) The Collapse of Confirm. *InformationWeek*, 19/10/1992, pp. 12-14.

Microsoft (2001) *Microsoft Announces Shared Development Process for Cooperation On Key Technology Initiatives.* Accessed on 28/11/2002 at: http://www.microsoft.com/presspass/press/2001/Jun01/06-19SDPpr.asp.

Microsoft (2003) *The Microsoft Solutions Framework on TechNet.* Accessed on 13/05/2003 at: http://www.microsoft.com/technet/itsolutions/tandp/innsol/default.asp.

Mintzberg, H. (1973) *The Nature of Managerial Work.* Harper & Row Publishers.

Mitev, N. N. (1996) More than a failure? The computerised reservation systems at French Railways. *Information Technology & People*, vol. 9, no. 4, pp. 8-19.

MMC (1993) *Gas: Volume 1 of reports under the Fair Trading Act 1973 on the supply within Great Britain of gas through pipes to tariff and non-tariff customers, and the supply within Great Britain of the conveyance or storage of gas by public gas suppliers. Monopolies and Mergers Commission.* Accessed on 19/06/2003 at: http://www.competition-commission.org.uk/reports/334gas.htm#full.

Mockus, A., Fielding, R. T. and Herbsleb, J. (2000) *A Case Study of Open Source Software Development: The Apache Server.* Accessed on 12/03/2003 at: http://opensource.mit.edu/papers/mockusapache.pdf.

Morisio, M., Stamelos, I. and Tsoukias, A. (2002) *A New Method to Evaluate Software Artifacts Against Predefined Profiles*. Accessed on 29/10/2003 at: http://11.lamsade.dauphine.fr/~tsoukias/papers/SEKE02-MorisioStamelosTsoukias.pdf.

Morris, P. W. G. and Hough G. H. (1987) *The Anatomy of Major Projects*. John Wiley & Sons.

Morris, P. W. G. (1997) *The Management of Projects*. Paperback Edition, Thomas Telford.

Morris, P. W. G. (2000) Researching the Unanswered Questions of Project Management. In *Proceedings of PMI Research Conference 2000, 21-24 June 2000, Paris, France*. Project Management Institute, pp. 87-101.

Mumford, E. (1983a) *Designing Participatively*, Manchester Business School.

Mumford, E. (1983b) *Designing Human Systems*, Manchester Business School.

Mylonopoulos, N. A., Doukidis, G. J. and Giaglis, G. M. (1995) *Information systems investment evaluation through simulation: the case of EDI*. Accessed on 16/07/2003 at: http://www.eltrun.aueb.gr/papers/edi5.html.

Mylopoulos, J. (1999) *Requirement-driven software development*. Accessed on 29/10/2003 at: http://www.sts.tu-harburg.de/projects/EUCAN/Workshop/09-99-Mylopoulos-Toronto-EUCAN.pdf.

NAO (2000) *The Cancellation of the Benefits Payment Card Project*. The National Audit Office, HC 857 Session 1999-2000. Accessed on 24/10/2001 at: http://www.nao.gov.uk/publications/nao_reports/9900857.pdf.

NAO (2001) *NIRS 2: Contract extension*. The National Audit Office, HC 355, Session 2001-2002. Accessed on 24/10/2001 at: http://www.nao.org.uk/publications/nao_reports/01-02/0102355.pdf.

NAO (2003) *New IT systems for Magistrates' Courts: the Libra project*. The National Audit Office, HC 327 Session 2002-2003. Accessed on 31/01/2003 at: http://www.nao.gov.uk/publications/nao_reports/02-03/0203327.pdf.

Nash, K. S. (2000) *Companies Don't Learn From Previous IT Snafus*. Accessed on 04/11/2002 at http://www.computerworld.com/printthis/2000/0,4814,53014,00.html.

Nathan Associates (1997) *Building an Information Economy — Software Industry Positions U.S. for New, Digital Era*. Nathan Associates Inc. Accessed on 12/11/2002 at: http://www.bsa.org/usa/globallib/econ/econstudy.pdf.

Nau, D. S. (1995) *Mixing Methodologies: Can Bimodal Research be a Viable Post-Positivist Tool?* Accessed 01/05/2002: http://www.nova.edu/ssss/QR/QR2-3/nau.html.

Naur, P. and Randell, B. (eds.) (1969) *Software Engineering: Report of a conference sponsored by the NATO Science Committee*, Garmisch, Germany, 7-11 Oct. 1968, Brussels. Accessed on 08/05/2003 at: http://homepages.cs.ncl.ac.uk/brian.randell/NATO/.

Nayar, P. (2000) *Ten Misconceptions about Software Documentation*. Accessed on 01/09/2003 at: http://www.stc.org/intercom/PDFs/2000/200005_27-29.pdf.

Newell, M. (1998) Communication Risk Management in Municipal Government Projects: City of New Orleans Computer-Aided Dispatch System Project. In D. I. Cleland, K. M. Bursic, R. Puerzer and A. Y. Vlasak (eds.), *Project Management Casebook*. Project Management Institute, pp. 85-98.

Nielsen, J. (1993) *Usability Engineering*. Academic Press.

NIH (2000) *Business Case For A National Institutes Of Health Business System*. Accessed on 08/09/2003 at: http://nbs.nih.gov/pdf/business_case.pdf.

OASIG (1996) *The performance of Information Technology and the role of human and organizational factors: Report to the Economic and Social Research Council UK*. Accessed on 07/01/2002 at: http://www.shef.ac.uk/~iwp/publications/reports/itperf.html.

Oberndorf, P. and Carney, D. (1998) *A Summary of DoD COTS-Related Policies*. Accessed on 28/08/2003 at: http://www.sei.cmu.edu/cbs/papers/monographs/dod-cots-policies/dod-cots-policies.pdf.

Offer (1997) *Review of Utility Regulation: submission by the Director General of Electricity Supply*. Accessed on 10/07/2003 at: http://www.ofgem.gov.uk/temp/ofgem/cache/cmsattach/2299_rursub.pdf.

Ofgem (1998) *Securing effective competition in the gas metering and meter reading services: The Director General's initial proposals*. Accessed on 19/06/2003 at: http://www.ofgem.gov.uk/temp/ofgem/cache/cmsattach/3339_meterunb.pdf.

Ofgem (2000a) *Final proposals for output measures and monitoring delivery between reviews*. Accessed on 10/07/2003 at: http://www.ofgem.gov.uk/temp/ofgem/cache/cmsattach/1606_iipout.pdf.

Ofgem (2000b) *Information and Incentives Project Draft Regulatory Instructions and Guidance*. Accessed on 10/07/2003 at: http://www.ofgem.gov.uk/temp/ofgem/cache/cmsattach/1584_iandirig.pdf.

Ofgem (2000c) *Securing Effective Competition In Gas Metering And Meter Reading Services: The Director General's final proposals*. Accessed on 19/06/2003 at: http://www.ofgem.gov.uk/temp/ofgem/cache/cmsattach/1549_meterdoc.pdf.

Ofgem (2001a) *Ofgem Embarks On Second Phase Of Information And Incentives Project*. Accessed on 10/07/2003 at: http://www.ofgem.gov.uk/temp/ofgem/cache/cmsattach/1050_r0801_18jan.pdf.

Ofgem (2001b) *Information And Incentives Project: Incentive Schemes - Initial Thoughts*. Accessed on 10/07/2003 at: http://www.ofgem.gov.uk/temp/ofgem/cache/cmsattach/2113_iip_incentives_intial.pdf.

Ofgem (2001c) *Information and Incentives Project Regulatory Instructions and Guidance*. Accessed on 10/07/2003 at: http://www.ofgem.gov.uk/temp/ofgem/cache/cmsattach/200_8feb01.pdf.

Ofgem (2001d) *Information and incentives project Incentive schemes: Final proposals*. Accessed on 10/07/2003 at: http://www.ofgem.gov.uk/temp/ofgem/cache/cmsattach/189_19dec01.pdf.

Ofgem (2002a) *The Regulation of Independent Gas Transporter Charging: Consultation document*. Accessed on 10/07/2003 at: http://www.ofgem.gov.uk/temp/ofgem/cache/cmsattach/382_16may02.pdf.

Ofgem (2002b) *Information and Incentives Project Regulatory Instructions and Guidance* version 2. Accessed on 10/07/2003 at: http://www.ofgem.gov.uk/temp/ofgem/cache/cmsattach/192_1march02_pub.pdf.

Ofgem (undated) *Industry Milestones*. Accessed on 01/09/2003 at: http://www.ofgem.gov.uk/ofgem/about/milestone.jsp.

OGC (2001) *OGC Gateway Process – Launch*. Accessed on 07/05/2003 at: http://www.ogc.gov.uk/index.asp?docid=750.

OGC (2002) *Forming Partnering Relationships*. The Office of Government Commerce. Accessed on 01/01/2003 at: http://www.ogc.gov.uk/index.asp?id=2894.

Olson, T. G., Gates, L. P., Mullaney, J. L. et al. (1993) *A Software Process Framework for the SEI Capability Maturity Model: Repeatable Level*. Accessed on 30/01/2003 at: http://www.sei.cmu.edu/publications/documents/93.reports/93.sr.007.html.

O'Neil, D. (1999) *Software Is Research...A Process Of Experimentation*. Accessed on 16/07/2003 at: http://members.aol.com/oneilldon2/competitor3-1.html.

O'Neill, D. (1996) *Regulated Industries: The UK Gas Industry*. Chartered Institute of Public Finance and Accountancy.

Orlikowski, W. J. (1993) CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development. Accessed on 02/05/2002 at: http://misq.org/archivist/bestpaper/misq93.html.

OSI (2001) *Advocacy*. Open Source Initiative. Accessed on 12/03/2002 at: http://www.opensource.org/advocacy/.

OSI (2002) *The Open Source Definition*. Open Source Initiative. Accessed on 28/11/2002 at: http://www.opensource.org/docs/definition.php.

Oz, E. (1994) When Professional Standards are Lax: The CONFIRM Failure and its Lessons. *Communications of the ACM*, vol. 37, no. 10, pp. 29-36.

Page, D., Williams, P. and Boyd, D. (1993) *Report of the Inquiry into the London Ambulance Service*. South West Thames Regional Health Authority.

Parker, D. and Hartley, K. (1997) The Economics of Partnership Sourcing vs Adversarial Competition. *European Journal of Purchasing and Supply Management*, vol. 3, no. 2, pp. 115-125.

Parker, D. and Kirkpatrick, C. (2002) *Researching Economic Regulation In Developing Countries: Developing A Methodology For Critical Analysis*. CRC International Workshop, 4-6 September 2002, Crawford House Lecture Theatre, University of Manchester. Accessed on 03/12/2002 at: http://idpm.man.ac.uk/crc/downloads/dparker.pdf.

Parker, R. P. and Grimm, B. (2000) *Recognition of Business and Government Expenditures for Software as Investment: Methodology and Quantitative Impacts, 1959-98*. Accessed on 12/11/2002 at: http://www.bea.doc.gov/bea/papers/software.pdf.

Parkin, M., Powell, M. and Mathews, K. (1997) *Economics*. 3rd Edition, Addison-Wesley.

Paulk, M. C. (1997) *Software Process Proverbs*. Accessed on 28/03/2002 at: http://www.stsc.hill.af.mil/crosstalk/frames.asp?uri=1997/01/proverbs.asp.

Pavlicek, R. C. (2000) *Embracing Insanity: Open Source Software Development*. Sams Publishing.

Perks, M. (2003) *Best Practices for Software Development Projects*. Accessed on 28/08/2003 at: http://www7b.software.ibm.com/wsdd/library/techarticles/0306_perks/perks2.html.

Pfleeger, S. L. (1998) *Software Engineering: Theory and Practice*. Prentice Hall.

Phel (undated) *Glossary of Terms: PFI*. Public Health Electronic Library. Accessed on 15/01/2003 at: http://www.phel.gov.uk/glossary/glossaryAZ.asp?getletter=P.

Phillips, E. M. and Pugh, D. S. (1994) *How to get a PhD: a handbook for students and their supervisors*. 2nd Edition. Open University Press.

Pich, M. T., Loch, C. H. and Meyer, A. D. (2002) On Uncertainty, Ambiguity and Complexity in Project Management. *Management Science*, vol. 48, no. 8, pp. 1008-1023. Accessed on 29/08/2003 at: http://www.insead.fr/~loch/articles/PM_MS_2002.pdf.

Pike, R. (2000) *Systems Software Research is Irrelevant*. Accessed on 15/02/2001 at: http://odin-os.sourceforge.net/download/os-research.pdf.

PMI (2000) *A Guide to the Project Management Body of Knowledge (PMBOK® Guide): Excerpts*. Project Management Institute. Accessed on 15/10/2002 at: http://www.pmi.org/prod/groups/public/documents/info/PP_PMBOKGuide2000Excerpts.pdf.

Pomberger, G. and Blaschek, G. (1996) *Object-Orientation and Prototyping in Software Engineering*. Prentice Hall.

Poppendieck, M. (2002) *Wicked Problems*. Accessed on 09/08/2002 at http://www.sdmagazine.com/print/documentID=24982.

Popper, K. R. (1959) *The Logic of Scientific Discovery*. Hutchinson.

Popper, K. R. (1963) *Conjectures and Refutations*. Routledge.

Porter, M. E. (1985) *Competitive Advantage: Creating and Sustaining Superior Performance*. The Free Press

Pressman, R. and Ince, D. (2000) *Software Engineering: a Practitioner's Approach*. Schaum.

Psillos, S. (1996) *Ampliative reasoning: induction or abduction?* Accessed on 24/5/2002 at: http://www.cs.bris.ac.uk/~flach/ECAI96/papers.html.

PWC (1999) *Contributions of the Packaged Software Industry to the Global Economy.* Accessed on 12/11/2002 at: http://www.bsa.org/usa/globallib/econ/pwc1999.pdf.

Raybould, A. (2002) *Process improvement alone is not the "Silver Bullet".* Accessed on 16/07/2003 at: http://news.cmcrossroads.com/articles/araug02.shtml.

Raymond, E. S. (1999) *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary.* O'Reilly & Associates.

Reece, M. (1999) *What is design management and what impact does it have on corporations?* Accessed on 16/07/2003 at: http://www.geocities.com/des_mgt_forum/what.html#anchor158762.

Remenyi, D. (1999) *Stop IT Project Failure Through Risk Management.* Butterworth-Heinemann.

Remenyi, D. (2000) The taming of the IT shrew - Delivering benefits from IT. *Southern African Business Review*, vol. 4, no. 1, pp. 28-35.

Remenyi, D., Williams, B., Money, A. and Swartz, E. (1998) *Doing Research in Business & Management, an introduction to process and method.* Sage Publications.

Roetzheim, W. H. (1993) Managing Software Projects: Unique Problems and Requirements. In P. C. Dinsmore (ed.), *The AMA Handbook of Project Management.* Amacom, pp. 347-352.

Robinson, C. (1996) Profit, Discovery and the Role of Entry: The Case of Electricity. In M. E. Beesley (ed.), *Regulating Utilities: Time for Change?* The Institute of Economic Affairs, pp. 109-140.

Robinson, W. S. (2000) The logical structure of Analytic Induction. (Originally published in *American Sociological Review*, vol. 16, no. 6, 1951, pp. 812-18). In R. Gomm, M. Hammersley and P. Foster (eds.), *Case Study Methods.* Sage Publications, pp. 187-195.

Robson, C. (1993) *Real World Research: a Resource for Social Scientists and Practitioner-Researchers.* Blackwell.

Rodrigues, A. and Bowers, J. (1996) The role of system dynamics in project management. *International Journal of Project Management,* vol. 14, no. 4, pp. 213-220.

Rodrigues, A. and Williams, T. (1997) System Dynamics In Software Project Management: Towards The Development Of A Formal Integrated Framework. *European Journal of Information Systems*, vol. 6, no. 1, pp. 51-66.

Rodrigues, A. and Williams, T. (1998) System Dynamics In Project Management: Assessing The Impacts Of Client Behaviour On Project Performance. *Journal of the Operational Research Society*, vol. 49, no. 1, pp. 2 – 15.

Roetzheim, W. H, (1993) Managing Software Projects: Unique Problems and Requirements. In P. Dinsmore (ed.), AMA Handbook of Project Management, AMACOM, pp. 347-352.

Rosen, G. A. (1999) *The Problem of Induction.* Accessed on 31/05/2002 at: http://www.princeton.edu/~grosen/puc/phi203/induction.html.

Rosenau, M. D. (1998) *Successful Project Management: a step-by-step approach with practical examples.* 3rd Edition. John Wiley & Sons.

Rosenberg, D. and Scott, K. (1997) *XP: Cutting Through the Hype.* Accessed on 16/07/2003 at: http://www.ratio.co.uk/ov3.pdf.

Rosenblatt, B. (1997) *Death March author Ed Yourdon admits he was wrong.* Accessed on 15/05/2002 at http://sunsite.uakom.sk/sunworldonline/swol-07-1997/swol-07-bookshelf.html.

Rowlinson, S. M. (1988) *An Analysis Of Factors Affecting Project Performance In Industrial Building.* PhD Thesis, Brunel University. Accessed on 25/04/2003 at: http://hkusury2.hku.hk/steve/phd.htm.

Rowlinson, S. M. and Lee, J. K. M. (2000) *Design Build - Some Problems In Implementation.* (Abstract). Accessed on 29/10/2003 at: http://www2.ing.puc.cl/~icccon/abstracts/PDF/Track4/T4-P6.pdf.

Royce, Winston (1970) Managing the Development of Large Software Systems. In *Proceedings of IEEE WESCON*, August 1970, pp. 1-9.

Royce, Walker (1998) *Software Project Management: A Unified Framework.* Addison-Wesley.

Royce, Walker (2002) *The Case For Results-Based Software Management.* Accessed on 15/11/2002 at: http://www.informationweek.com/story/IWK20020517S0002.

Royer, I. (2003) Why Bad Projects Are So Hard To Kill. *Harvard Business Review*, vol. 81, no. 2, pp. 48-56.

Ruiz, M., Ramos, I. and Toro, M. (2001) A simplified model of software project dynamics. *Journal of Systems and Software*, vol. 59, no. 3, pp. 299-309.

Russo, N. and Stolterman, E. (2000) Exploring the assumptions underlying information systems methodologies: their impact on past, present and future ISM research. *Information Technology and People*, vol. 13, no. 4, pp. 313-327.

Sappington, D. E. M. (1991) Incentives in Principal-agent Relationships. *Journal of Economic Perspectives*, vol. 5, no. 2, pp. 45-66.

Sauer, C. (1993) Partial Abandonment as a Strategy for Avoiding Failure. *IFIP Transactions A – Computer Science and Technology*, vol. 24, pp. 143-167.

Saunders, M., Lewis, P. and Thornhill, A. (2000) *Research Methods for Business Students.* 2nd Edition. Pearson Education.

Sawyer, S. (2000) Packaged software: implications of the differences from custom approaches to software development. *European Journal of Information Systems*, vol. 2000, no. 9, pp. 47–58. Accessed on 18/11/2002 at: http://www.ucab.edu.ve/postgrado/sist-info/info-org/materiales/pcksoft.pdf.

Scacchi, W. (2002) Understanding the Requirements for Developing Open Source Software Systems. *IEE Proceedings--Software*, vol. 149, no. 1, pp. 24-39.

Schulte, R. (2002) *Modern Cost Management.* Accessed on 29/08/2003 at: http://www.projectmagazine.com/v3i8/costv3i8.html.

Schwaber, K. (1996), *SCRUM Development Process.* Accessed on 29/08/2003 at: http://jeffsutherland.com/oopsla/schwapub.pdf.

Shattock, M. (2001) Capsa And Its Implementation Report To The Audit Committee And The Board Of Scrutiny: Part B. *Cambridge University Reporter*, no. 5861, 2 November 2000.

Shaw, M. and Garlan D. (1996) *Software Architecture: Perspectives on an Emerging Discipline.* Prentice Hall.

Shortliffe, E. H. and Perreault, L. E. (eds.) (2000) *Medical Informatics: Computer Applications in Health Care and Biomedicine.* Springer-Verlag.

Smit, J. and Bryant, A. (2000) *Grounded Theory Method in ISD research: Glaser vs. Strauss.* Accessed on 30/05/2002 at: http://www.lmu.ac.uk/ies/im/Research/2000-7.pdf.

Smith, C. and Tonks, S. (1995) Partnering. In J. R. Turner (ed.), *The Commercial Project Manager,* McGraw-Hill, pp. 119-129.

Smith, J. M. (2001) *Troubled IT Projects: Prevention and Turnaround.* The Institute of Electrical Engineers.

Smith, N. C. (1991) The case-study: a vital yet misunderstood research method for management. In N. C. Smith and P. Dainty (eds.) *The Management Research Handbook,* Routledge, pp. 145-158.

Sobek, D. K., Ward, A. C. and Liker, J. K. (1999) Toyota's principles of set-based concurrent engineering. *Sloan Management Review,* vol. 40 (winter), pp. 67–83.

Soderlund, J. (2004) Building theories of project management: past research, questions for the future. *International Journal of Project Management,* vol. 22, pp. 183–191.

Sol, S. (1998) The Open Source Business Model! Accessed 27/02/2003 at: http://www.wdvl.com/Software/Open/Source/business_model.html.

Southon, G., Sauer, C. and Dampney, C. N. G. (1997) Information Technologies in Complex Health Services: Organizational Impediments to Successful Technology Transfer and Diffusion. *Journal of the American Medical Informatics Association,* vol. 4, pp. 112-124.

Spendolini, M. J. (1992) *The Benchmarking Book.* Amacom.

SPR (2002) *Choosing Function Points or Feature Points.* Software Productivity Research. Accessed on 03/11/2003 at: http://www.spr.com/products/choosing.htm.

Spradley, J. P. (1980) *Participant Observation.* Holt, Rinehart and Winston.

SRA (2002) *Ethical Guidelines 2002.* Social Research Association. Accessed on 07/05/2002 at: http://www.the-sra.org.uk/ethics02.pdf.

Stake, R. E. (1994) Case Studies. In N. Denzin and Y. Lincoln (eds.) *The Handbook of Qualitative Research,* Sage Publications, pp. 236-247.

Standish Group (1995): *Chaos Report.* Accessed on 15/10/2002: http://www.pm2go.com/sample_research/chaos_1994_1.php.

Stapleton, J. (1997) *Dynamic Systems Development Method: The Method In Practice.* Addison-Wesley.

Steiner, I. D. (1966) Models for inferring relationships between group size and potential group productivity. *Behavioral Science*, vol. 5, no. 11, pp. 273-283.

Stephens, M. (2003) *The Case Against Extreme Programming.* Accessed on 16/07/2003 at: http://www.softwarereality.com/lifecycle/xp/case_against_xp.jsp.

Sterman, J. (2000) *Business dynamics: systems thinking and modelling for a complex world.* McGraw-Hill.

Stewart, A. (2002) The Death of Moore's Law? *Computer Headline,* Issue 79, February.

Strauss, A. L. (1987) *Qualitative Analysis For Social Scientists.* Cambridge University Press.

Strauss, A. and Corbin, J. (1990) *Basics Of Qualitative Research.* Sage Publications.

Strauss, A. and Corbin, J. (eds.) (1997) *Grounded Theory in Practice.* Sage Publications.

Suzuki, Y. (2002) *Managed Competition as an Incentive Mechanism in Supply Relations.* Accessed on 23/05/2003 at: http://www2.mt.tama.hosei.ac.jp.

Tashakkori, A. and Teddlie, C. (1998) *Mixed Methodology, Combining Qualitative And Quantitative Approaches*. Sage Publications.

Teach, E. (2000) *Our Bond Is Your Bond.* Accessed on 15/01/2003 at: http://www.cfo.com/Article?article=1717.

TechTarget (2001) *Industrial Strength.* Accessed on 08/10/2003 at: http://whatis.techtarget.com/definition/0,,sid9_gci212340,00.html.

TechTarget (2002a) *Waterfall Model.* Accessed on 15/10/2002 at: http://searchvb.techtarget.com/sDefinition/0,,sid8_gci519580,00.html.

TechTarget (2002b) *Flash.* Accessed on 15/10/2002 at: http://whatis.techtarget.com/definition/0,,sid9_gci214563,00.html.

Thagard, P. R. (1978) The Best Explanation: Criteria for Theory Choice. *The Journal of Philosophy*, vol. 75, no. 2, pp. 76-92.

Thomas, A. J. (2000) *Project Manager's Desk Reference: Guide to Project Management Body of Knowledge.* The Hampton Group.

Ticehurst, G. W. and Veal, A. J. (2000) *Business Research Methods, a management approach.* Longman.

Trochim, W. M. (2002) *Research Methods Knowledge Base.* Accessed on 02/05/2002 at: http://trochim.omni.cornell.edu/kb/.

Trusted, J. (1987) *Inquiry and Understanding: An Introduction to Explanation in the Physical and Human Sciences.* MacMillan Education.

Turner, J. R. and Cochrane, R. A. (1993) Goals-and-methods matrix: coping with projects with ill defined goals and/or methods of achieving them. *International Journal of Project Management*, vol. 11, no. 2, pp. 93-102.

Underhill, E. (2002) *The Australian Construction Industry: Union Control In A Disorganised Industry.* Accessed on 08/09/2003 at: http://www.business.vu.edu.au/mgtcontent/Working Paper.pdfs/Elsa Underhill 6-2002.pdf.

Urquhart, C. (2000) Strategies For Conversation And Systems Analysis In Requirements Gathering: A Qualitative View Of Analyst-Client Communication. *The Qualitative Report*, vol. 4, no. 1. Accessed on 30/05/2002 at: http://www.nova.edu/ssss/QR/QR4-1/urquhart.html.

Varian, H. R. (2001) *Economics of Information Technology.* Accesses on 30/12/2002 at: http://www.sims.berkeley.edu/~hal/Papers/mattioli/mattioli.html.

Vienneau, R. (1996) *Software Crisis or Hardware Success.* Accessed on 08/09/2003 at: http://192.73.45.130/awareness/newsletters/summer96/crisis.success.html.

Vitharana, P. and Zahedi, F. (1997) *Group Decision Support for Software Requirements Analysis.* Accessed on 17/07/2003 at: http://hsb.baylor.edu/ramsower/ais.ac.97/papers/vith_zah.htm.

Vowler, J. (1999) *Swimming Against The Tide.* Accessed on 03/01/2002 at: http://www.findarticles.com/cf_dls/m0COW/1999_Jan_14/53624854/p1/article.jhtml.

Waller, A. (2002) *Measuring the benefits of information systems.* Accessed on 29/08/2003 at: http://www.remitconsulting.com/articles/article2002June.htm.

Walsh, K. R., and Schneider, H. (2002) The role of motivation and risk behaviour in software development success. *Information Research*, vol. 7, no. 3. Accessed on 24/02/2003 at: http://InformationR.net/ir/7-3/paper129.html.

Ward, J. A. (2001) *The Key Project Constraints: Relationships and Tradeoffs.* Accessed on 11/07/2003 at: http://www.jamesaward.com/constraints.doc.

Ward, J. and Peppard, J (2002) *Strategic Planning for Information Systems.* 3rd Edition. John Wiley & Sons.

Wass V. J. and Wells, P. E. (eds.) (1994) *Principles and Practice in Business and Management Research.* Dartmouth Publishing Company.

Webster, B.F. (2000) *Patterns In IT Litigation: Systems Failure (1976-2000) A Study By Pricewaterhousecoopers.* Accessed on 09/12/2002 at: http://www.bfwa.com/litigate/sysfail-bfw.doc.

Weinberg, G. (1971) *The Psychology of Computer Programming.* Van Nostrand-Reinhold.

Wheatley, M. (2000) *ERP Training Stinks.* Accessed on 17/01/2003 at: http://www.cio.com/archive/060100/erp.html.

Whitten, N. (1995) *Managing Software Development Projects: Formula for Success.* 2nd Edition. John Wiley & Sons.

Wideman, R. M. (2001) *The Future of Project Management.* Accessed on 24/01/2002 at: http://www.maxwideman.com/papers/index.htm.

Wiegers, K. (1998) *Know Your Enemy: Software Risk Management.* Accessed on 28/08/2003 at: http://www.processimpact.com/articles/risk_mgmt.pdf.

Wiegers, K. (2003) *See You in Court.* Accessed on 28/08/2003 at: http://www.processimpact.com/articles/court.html.

Williams, L. and Kessler, R. (2002) *Pair Programming Illuminated.* Addison-Wesley.

Williams, L., Kessler, R. R., Cunningham, W., and Jeffries, R. (2000). Strengthening the case for pair programming. *IEEE Software,* vol. 17, no. 4, pp. 19-25.

Williams, L. and Cockburn, A. (2003) *Agile Software Development; It's about Feedback and Change.* Accessed on 21/08/2003 at: http://www.computer.org/computer/homepage/0603/GEI/r6039.pdf.

Williams, T. (1995) A classified bibliography of research relating to project risk. *European Journal of Operational Research,* vol. 85, pp. 18-38.

Williams, T. (2003a) Learning from projects. *Journal of the Operational Research Society,* vol. 54, no. 5, pp. 443-451.

Williams, T. (2003b) Identifying the hard lessons from projects – easily. *International Journal of Project Management,* Article in Press (available online 18 December 2003).

Williford, J. and Chang, A. (1999) Modelling the FedEx IT division: a system dynamics approach to strategic IT planning. *Journal of Systems and Software,* vol. 46, no. 2-3, pp. 203-211.

Wilson, G. V. (1998) *Dancing Around the Problem.* Accessed on 27/05/2003 at: http://www.ercb.com/feature/feature.0030.html.

Wolinsky, A. (1997) Regulation of Duopoly: Managed Competition vs Regulated Monopolies. *Journal of Economics & Management Strategy,* vol. 6, no. 4, pp. 821-847.

Wright, D. (2001) *Worldwide IT Spending 2001-2005: Forecast and Analysis for Hardware, Software, and Services.* Accessed as of 12/03/2002 at: http://www.aberdeen.com/ab%5Fcompany/hottopics/itspending/thankyou.htm.

Yardley, D. (2002) *Successful IT Project Delivery: Learning the Lessons of Project Failure.* Pearson Education.

Yeargin, R. (2002) *Why software sucks.* Accessed on 08/09/2003 at: http://librenix.com/?inode=2068.

Yin, R. K. (1994) *Case study research: Design and methods.* 2nd Edition, Saga Publications.

Yourdon, E. (1995) *A Reengineering Concept for IT Organizations: "Good-Enough" Software.* Accessed on 29/10/2003 at: http://www.yourdon.com/articles/GoodEnuf.html.

Yourdon, E. (1997) *Death March: The Complete Software Developer's Guide to Surviving 'Mission Impossible' Projects.* Prentice Hall.

Yourdon, E. (2002) *Preface to "Better Software Faster, a book by A. Carmichael and D. Haywood.* Accessed on 12/06/2003 at: http://www.mycgiserver.com/~bswf/foreword.htm.

Zimmerman, M. (2000) Weapon System Competitive "Shoot-off". In A. Laufer and A. Hoffman (eds.), *Project Management Success Stories.* John Wiley & Sons, pp. 65-67.

Zimmerman, C. (2001) *Examination Of Open-Source Software Development.* Accessed on 27/02/2003 at: http://www.zimwiz.com/research.

Ziv, H., Richardson, D. J. and Klosch, R (1996) *The Uncertainty Principle In Software Engineering.* Accessed on 28/08/2003 at: http://citeseer.nj.nec.com/ziv96uncertainty.html.

Appendix A: Acronyms Used in The Thesis

AIM	Application Implementation Methodology
AR	Action Research
ASP	Active Server Page
BCL	Bought-In Contractor Labour
BPR	Business Process Re-Engineering
CAD	Computer Aided Dispatch
CBS	Customer Billing Systems
CBSW	Custom-Built Software
CMM	Capability Maturity Model
COM	Common Object Model
COTS	Commercial-Off-The-Shelf
CRS	Customer Registration Systems
CRS	Computerised Registration System
CS2000	Commerce Server 2000
DLC	Direct Labour Cost
DSDM	Dynamic Systems Development Method
EAC	Estimated At Completion
ERP	Enterprise Resource Planning
FAT	Factory Acceptance Test
GAO	General Accounting Office
GCMR	Gas Customer Meter Reading (System)
GDCS	Gas Data Collection System
GMRM	Gas Meter Reading Management
GOTS	Government Off-The-Shelf

GUI	Graphic User Interface
HHT	Hand-Held Terminal
HV	High Voltage
IBE	Inference To The Best Explanation
IDL	Internal Direct Labour
IIP	Information And Incentive Project
IPR	Intellectual Property Rights
IS	Information System
ISD	Information System Development
ISDM	Information System Development Methodology
ISM	Information System Methodology
IT	Information Technology
ITT	Invitation To Tender
LAS	London Ambulance Service
LV	Low Voltage
LVC	LV Connectivity
MOTS	Military Off-The-Shelf
MPAN	Metering Point Administration Number
MPS	MRA/PGT Separation (Project)
MRA	Meter Reading Agency
MRS	Meter Reading Services
MWMS	Meter Work Management System
NAO	National Audit Office
NAPF	Non-Acceptance Price Factor
NDM	Non Daily Meter
Offer	Office Of Electricity Regulation
Ofgem	Office Of Gas And Electricity Market

OMS	Outage Management System
OODM	Object-Oriented Design Methodology
OSS	Open Source Software
PA	Principal-Agent
PBSDF	Performance-Based Systems Development Framework
PBSDF-MSS	PBSDF Multiple Supplier Strategy
PBSDF-SSS	PBSDF Single Supplier Strategy
PBSDF-TSS:	PBSDF Two Supplier Strategy
PCR	Project Charge Rate
PDD	Product Delivery Date
PES	Public Electricity Supply
PGT	Public Gas Transporter
PIR	Project Issues Report
PMI	Project Management Institute
PPI	Propensity For Price Increase
PPSW	Pre-Packaged Software
QAT	Quality Assurance Team
QRAM	Quantitative Risk Analysis Matrix
RAD	Rapid Application Development
RFP	Request For Proposal
RFQ	Request For Quotation
RHA	Regional Health Authority
RISP	Regional Information Systems Plan
RM	Release Manager
RMT	Requirements Management Team
RPE	Relative Product Evaluation

RPM	Relative Performance Measurement
SAT	Site Acceptance Test
SBCE	Set Based Concurrent Engineering
SCP	Structure, Conduct And Performance
SD	Systems Dynamics
SDLC	Systems Development Lifecycle
SLAT	System Level Architecture Team
SLMP	System Level Manufactured Products
SMT	Supplier Management Team
SOW	Statement Of Work
SPMS	Supplier Point Management System
SPR	Software Productivity Research
SQL	Sequential Query Language
SRA	Social Research Association
SSADM	Structured System Analysis And Development Methodology
T&M	Time And Materials
TQM	Total Quality Management
TSDF	Traditional Systems Development Framework
TSS	Two Supplier Strategy (PBSDF)
UMRS	Unbundled Meter Reading System
VE	Value Engineering
VM	Value Management
WBS	Work Breakdown Structure
XP	eXtremeProgramming

Appendix B: 40 Root Causes Compiled by (Smith, 2001, pp. 18-19)

(See Section 3.4 for comments on this catalogue of root causes)

Project Conception	
RC01	Project based on an unsound premise or an unrealistic business case
RC02	Buyer failure to define clear project objectives, anticipated benefits and success criteria
RC03	The Project is based on state-of-the-art and immature technology
RC04	Lack of Buyer board-level ownership/commitment or competence
RC05	The Buyer's funding and/or timescale expectations unrealistically low
RC06	Failure to break a complex project into phases or smaller projects
Project initiation/mobilisation	
RC07	Vendor setting unrealistic expectations on cost, timescale or vendor capability
RC08	Buyer failure to define and document requirements (functional and non-functional)
RC09	Failure to achieve an open, robust and equitable buyer-vendor relationship
RC10	Vendor failure to invest enough resources to scope the project prior to contract
RC11	Lack of sufficient involvement of eventual end users
RC12	Vendor underestimation of resources (predominantly person-effort) required
RC13	Vendor failure to define project tasks, deliverables and acceptance processes
RC14	Failure to actively manage risks and maintain robust contingency plans
RC15	Poor project planning, management and execution
RC16	Failure to clearly define roles and responsibilities in the contract/sub-contracts
RC17	Full-scope, fixed-price contracting (requirements, design and development)
System Design	
RC18	Failure to "freeze" the requirements baseline and apply change control
RC19	Poor choice of technical platform and/or architecture
RC20	Vendor starting a phase prior to completing a previous phase
RC21	Poor choice of design/development method
RC22	Failure to undertake effective project reviews and take decisive action
RC23	Vendor lack/loss of resources
RC24	Poor vendor standards deployment (design, coding, testing, configuration management, etc.)
RC25	Poor vendor requirements traceability (requirements - design - code - test)
RC26	Buyer retains design authority with right to approve/reject low-level designs

System development	
RC27	Delays cause the project to be overtaken by advances in technology
RC28	Vendor failure to "freeze" the design (and technical platform) and apply change control
RC29	Inadequate vendor training and supervision of junior staff
RC30	Inadequate vendor review of designs/code/documentation
RC31	Poor vendor management of subcontractors
RC32	Lack of a formal "engineering" approach to integration and testing by the vendor
RC33	Insufficient attention paid by vendor to non-functional requirements
System implementation	
RC34	Buyer failure to manage the change implicit in the project (people, processes, technology)
RC35	Inadequate user/systems training
RC36	Catastrophic failure of the system, with no effective contingency arrangement
RC37	Missing a crucial "go live" date
System operation, benefit delivery, stewardship and disposal	
RC38	Buyer failure to measure actual delivered benefit and take corrective action
RC39	Buyer failure to maintain/enhance system post-implementation
RC40	Changes in the competitive or macro-economic environment

Appendix C: Quantitative Risk Analysis Matrix (QRAM) Template

DESCRIPTION			PRE-MITIGATION ANALYSIS			MITIGATION			RESIDUAL RISK		
Risk Type	Priority	Specific Risk Item	Prob of Occur (%)	Cost to Correct (£)	Likely Impact (£)	Activity (What, Who, When)	Delta Cost (£)	Delta Time (days)	Prob of Occur (%)	Cost to Correct (£)	Remaining Impact (£)
Technical					£0					£0	£0
Schedule					£0					£0	£0
Contractual					£0					£0	£0
Programmatic					£0					£0	£0
Cost					£0					£0	£0
					£0		£0				£0

MANAGEMENT RESERVE:	£0