A Synthesised Approach to Goal-Oriented Requirements Engineering

by

Stewart John Marshall Green

A thesis submitted in part fulfilment of the requirements for the degree of Doctor of Philosophy in the faculty of Engineering of the University of London, and the Diploma of Imperial College of Science, Technology and Medicine.

April, 2004
Abstract

Requirements engineering for complex socio-technical systems has frequently proved difficult and costly to undertake; and has often resulted in inappropriate applications being built. Despite research in requirements engineering, problems still exist. A new family of approaches from different academic stables—goal-oriented requirements engineering, socio-technical systems development, and business process modelling—appears to offer potential for improving this situation. However, it has proved difficult to select and apply the appropriate parts of each of these approaches. To address this problem, a framework of requirements engineering method building principles and features has been created from a critical of these approaches. This framework has been used both to synthesise a new method, and to compare it to these new approaches. A large real-world case study was used to validate the new method. Critical evaluation has identified areas for further work on the new method.
Statement of Contribution

A framework of requirements engineering method building principles and features has been created from a critical review of three families of existing approaches to requirements engineering: socio-technical, goal-oriented, and process-oriented. Using this framework, a new goal-oriented approach to requirements engineering has been synthesised. Because the new approach is based upon principles underpinning the families of selected approaches, it is expected that it will help to solve some of the problems associated with early approaches to requirements engineering: poor understanding of the wider domain, insufficient attention paid to business goals, the wrong system selected for development, and systems frequently rejected by stakeholders, for example.

The approach is specified in detail. It is intended that this will encourage its widespread use in organisations by requirements engineering practitioners. To the extent that this occurs, it will be possible to evaluate the effectiveness of goal-based approaches, to determine when they are the best approach to take, and to compare their effectiveness to the effectiveness of other approaches to requirements engineering.
I would like to thank first my wife, Paik Sun, for her support in patiently waiting for me to complete this thesis. Next, I would like to thank Professor Anthony Finkelstein for his belief in the value of this work, and for his supervision. I would also like to acknowledge the contribution of Professor Jeff Kramer, who provided invaluable feedback in the role of second supervisor. During the course of this work, several colleagues helped me to overcome particular hurdles by reading the thesis and providing feedback. For these contributions, I would particularly like to thank Mr Ian Beeson, Dr Sara Jones, and Dr David Coward. I am also indebted to all of the stakeholders who gave up their time to participate in the case study; in particular, I would like to thank Mr Jon Ward, who acted as the first client. Finally, I would like to acknowledge a debt of gratitude to the Faculty of Computing, Engineering and Mathematics at the University of the West of England, for providing financial support and the time to allow me to undertake the research for this thesis.
# Contents

Abstract 2

Statement of Contribution 3

Acknowledgements 4

1 Introduction 12
   1.1 Background and motivation 13
   1.2 Statement of problem 14
   1.3 Contribution of thesis 14
   1.4 Novel features 16
   1.5 Thesis structure 17
   1.6 Aids for the reader 18

2 Requirements engineering 19
   2.1 Introduction 20
   2.2 Problems of software-intensive systems engineering 20
   2.3 The solution: requirements engineering 21
   2.4 Requirements engineering 22
   2.5 Problems of requirements engineering 23
   2.6 Summary 25

3 Requirements in context 26
   3.1 Introduction 27
   3.2 Socio-technical approaches to requirements engineering 27
   3.3 Problems of the socio-technical approaches 45
   3.4 Stakeholders 46
   3.5 Summary 47

4 Deriving requirements from goals 49
   4.1 Introduction 50
   4.2 Rationale for a goal-oriented approach to requirements engineering 50
   4.3 The goal-oriented approach to requirements engineering 51
   4.4 Goal-oriented approaches in the literature 55
   4.5 Advantages and problems of the goal-oriented approach 66
   4.6 Goals and processes 70
   4.7 Summary 80
## Contents

5 Relating the socio-technical and goal-oriented approaches  
5.1 Introduction .................................................. 82  
5.2 A discussion and generic model .............................. 82  
5.3 Early attempts at integration ................................. 84  
5.4 Summary ...................................................... 88  

6 A synthesised goal-oriented approach  
6.1 Introduction .................................................. 90  
6.2 Rationale for the synthesised approach ..................... 90  
6.3 The synthesised approach .................................. 95  
   6.3.1 Part-one: Conceptualise the current served system .. 95  
   6.3.2 Part-two: Select stakeholders ......................... 115  
   6.3.3 Part-three: Derive new requirements .................. 117  
6.4 Summary ...................................................... 131  

7 Case study  
7.1 Introduction .................................................. 134  
7.2 Rationale for using a case study ............................ 134  
7.3 The case study ............................................... 136  
   7.3.1 Part-one: Conceptualise the current served system .. 136  
   7.3.2 Part-two: Select stakeholders .......................... 169  
   7.3.3 Part-three: Derive new requirements .................. 174  
7.4 Summary ...................................................... 195  

8 Critical evaluation  
8.1 Introduction .................................................. 203  
8.2 Evaluation of the approach ................................ 203  
8.3 Evaluation of the contribution of the thesis ............... 210  
8.4 Summary ...................................................... 212  

9 Conclusion  
9.1 Introduction .................................................. 214  
9.2 Thesis summary .............................................. 214  
9.3 Main achievements .......................................... 214  
9.4 Future work .................................................. 216  

Bibliography ...................................................... 226  

Appendices  
A The Meeting Scheduler Problem: Preliminary Definition 228  
B Acronyms and glossary of terms .............................. 231  
C RAD notation .................................................. 233  
D Client questionnaire pro-forma ............................... 235  
E Subsystem owner questionnaire pro-forma ..................... 249  
F Case-study client’s responses to Client Questionnaire .... 254  
G Validate client’s view of the served system ................. 275
H Supporting the User Support/Problem Logging goal 283
I User questionnaire 286
J Analysis and review of users’ attitudes towards COTS 290
K Client’s feedback on the analysis of users’ attitudes to COTS 304
L The wider served system 307
M Subsystem owners’ responses to questionnaire 315
N Observing the COTS user-problem management system 341
O The flow of problems into and around the COTS domain 358
P Client’s view of goals for new served system 364
Q Other stakeholders’ views of goals for new served systems 377
R Tables of stakeholders’ problems, goals, and constraints 388
S Transformation of stakeholders’ problems into goals: 395
T Stakeholder Three’s Goal Hierarchy Diagram 400
U Report-1: long-term problems and goals for a future system 405
V The stakeholders’ response to Report 1 430
W Meeting 2: minutes 432
X Alternative designs satisfying the stakeholders’ goals 436
Y Stakeholder-Meeting-3 465
List of Figures

3.1 Checkland’s ISD process ........................................ 34
4.1 Goal decomposition (after [LK95a]) .......................... 52
4.2 Goal lattice - a fragment .......................................... 53
4.3 Air Traffic Control goal hierarchies (after [LK95b]) ....... 58
4.4 Air Traffic Control goal graph (after [LK95b]) ............ 59
4.5 Air Traffic Control functional requirements (after [LK95b]) 61
4.6 Air Traffic Control non-functional requirements (after [LK95b]) 61
4.7 Overview of the KAOS approach .............................. 62
4.8 A portion of a KAOS meta model .............................. 63
4.9 The KAOS process .................................................. 64
4.10 Partial decomposition of a meeting scheduler system goal .. 67
4.11 Operationalising goals into constraints ....................... 68
4.12 Fragment of a requirements model for a meeting scheduler system 69
4.13 Provision of IT support (one) .................................... 71
4.14 Provision of IT support (two) ..................................... 71
4.15 OPM: a RAD diagram model ................................. 77
4.16 OPM: system model for the Helpdesk context ............ 78
4.17 OPM: goal model for the Helpdesk context .............. 78
4.18 OPM: method model for the Helpdesk context ........... 79
4.19 OPM: creating the active model ............................. 80
5.1 A process model for goal-oriented requirements engineering .... 84
6.1 High-level view of the synthesised approach ............... 92
6.2 UML class diagram of served system features ............. 97
6.3 System Map class diagram ....................................... 98
6.4 System Map iconography ....................................... 99
6.5 System Map example ........................................... 99
6.6 System-subsystem Diagram class diagram .................. 100
6.7 System-subsystem Diagram iconography .................... 100
6.8 System-subsystem Diagram example .......................... 100
6.9 Client’s goals class diagram .................................... 102
6.10 Client’s goals iconography ..................................... 102
6.11 Client’s goals example .......................................... 102
6.12 Goals impacted by problems class diagram ............... 106
6.13 Goals impacted by problems iconography .................. 106
6.14 External Goals Client Goals Map ............................ 110
6.15 Characterisation in UML of Problem Goal ................. 122
6.16 Iconography of Problem Goal ............................... 122
6.17 Example of a Problem Goals .................................................. 122
6.18 Iconography of dependencies between Problem Goals ................. 123
6.19 Example of dependencies between Problem Goals ....................... 123
6.20 Characterisation in UML of Stakeholder Goal .......................... 125
6.21 Iconography of Stakeholder Goal ........................................ 125
6.22 Example of Stakeholder Goals ............................................. 125
6.23 Characterisation in UML of Inferred Goal ................................ 126
6.24 Iconography of Inferred Goal .............................................. 126
6.25 Example of an Inferred Goal ............................................... 127

7.1 System map showing boundary of client’s domain ........................ 138
7.2 Sub-divisions in the client’s domain ...................................... 138
7.3 Client’s goals .................................................................... 140
7.4 High-level goals assigned to sub-division manager ....................... 141
7.5 Impact of problems on goals ................................................ 143
7.6 External Goals Client Goals Map .......................................... 147
7.7 The resolve user-problem process (variation one) ......................... 163
7.8 The resolve user-problem process (variation two) ......................... 164
7.9 The system expert resolves user-problem process ......................... 165
7.10 The resolve long-term user-problem process ............................ 166
7.11 The old University user-problem management system ................. 171
7.12 Client’s problem goals ....................................................... 182
7.13 Client’s problem goals reorganised ...................................... 182
7.14 Client’s problem goals and goals ......................................... 183
7.15 Client’s Goal Hierarchy Diagram ......................................... 184
7.16 Stakeholder Three’s Goal Hierarchy Diagram .......................... 185
7.17 Emerging Composite Goal Hierarchy Diagram .......................... 187
7.18 Goal conflict over allowed channels of communication ............... 192
7.19 System Map of the new served system .................................. 201

C.1 The RAD notation .................................................................. 234

H.1 The resolve user-problem process ........................................... 285

T.1 Client’s problem goals .......................................................... 400
T.2 Client’s problem goals .......................................................... 401
T.3 Stakeholder Three’s Goal Hierarchy Diagram .......................... 402
T.4 Stakeholder Three’s Goal Hierarchy Diagram .......................... 404

U.1 The resolve user-problem process (variation one) ......................... 407
U.2 The resolve user-problem process (variation two) ......................... 408
U.3 The system expert resolves user-problem process ......................... 409
U.4 The resolve long-term user-problem process ............................ 409

X.1 COTS organisation: old ....................................................... 438
X.2 COTS organisation: new .................................................... 439
X.3 Goal conflict over recording short-term user-problem details ........... 444
X.4 Goal conflict over allowed channels of communication ............... 448
X.5 COTS-users communication protocol diagram .......................... 456
List of Tables

6.1 Client’s High-level Goals Table ......................................... 101
6.2 Domain Tasks Table ..................................................... 103
6.3 Domain Problems Table .................................................. 104
6.4 Goals Impacted By Problems Table .................................... 105
6.5 Desired improvements Table ............................................ 107
6.6 Goals impacted by improvements ....................................... 108
6.7 External Goals Table ..................................................... 109
6.8 Subsystem Owner’s High-level Goals Table ........................... 112
6.9 Client’s Goal Subsystem Owners’ Goals table ........................ 113
6.10 Client’s Problems Table ................................................ 118
6.11 Client’s Goals Table .................................................... 119
6.12 Client’s Constraints Table .............................................. 119
6.13 Stakeholder Goals Table ............................................... 120
6.14 Transform Problems To Goals Tables .................................. 121
6.15 Stakeholders’ goals and corresponding requirements ............. 132

7.1 Client’s high-level goals .................................................. 139
7.2 High-level goals assigned to sub-division managers ................ 141
7.3 Domain tasks identified by the client ................................ 142
7.4 Domain problems identified by the client ........................... 142
7.5 Goals impacted by problems ............................................ 143
7.6 Improvements to the domain identified as desirable by the client 144
7.7 Goals impacted by improvements ....................................... 144
7.8 External Goals Table for the Dean ..................................... 146
7.9 External Goals Table for the Service Level Agreement ............. 146
7.10 External Goals Table for the Principal Computing Officer ........ 147
7.11 Subsystem Owner’s High-level Goals Table .......................... 151
7.12 Goals linked to client’s goal Support Users .......................... 152
7.13 Goals linked to client’s goal Achieve System Availability Targets 153
7.14 Goals linked to client’s goal deploy Best Kit ....................... 154
7.15 Problem resolution ....................................................... 155
7.16 Long-term problems ...................................................... 156
7.17 Managing long-term problems: problems and improvements .... 157
7.18 Observation period problems and their resolution ................. 161
7.19 Summary of problems encountered in the observation period .... 161
7.20 Flow of problems into and around COTS: Tuesday ................ 167
7.21 Case study Client’s Problems Table ................................... 176
7.22 Case study Client’s Goals Table ....................................... 177
7.23 Case study Client’s Constraints Table ................................ 178
7.24 Case study: Stakeholder Three’s Problems Table .................. 179
7.25 Case study: Stakeholder Three’s Goals Table .......................... 180
7.26 Transforming the client’s problems to corresponding goals ....... 181
7.27 Merging Goal Hierarchy Diagrams: Decisions ......................... 186
7.28 Stakeholders’ goals and corresponding requirements ............... 196

8.1 Using the framework to compare the new and established ap-
proaches .............................................................................. 205

B.1 Acronyms ........................................................................... 231
B.2 Term definitions ................................................................. 232

J.1 User satisfaction with COTS services .................................... 293
J.2 User satisfaction with COTS services (continued) ................. 294
J.3 Problems perceived by users (ordered by number of comments) 296
J.4 Improvements to existing services suggested by users) .......... 297
J.5 New services suggested by users) ........................................ 298
J.6 User responses in the “service quality” category .................... 299
J.7 User responses in the “problem management” category ......... 300
J.8 User responses in the “communication” category ................. 302
J.9 User responses in the “printouts” category ......................... 303

N.1 Summary of problems encountered in the observation period .. 346
N.2 Observation period problems and their resolution ............... 351
N.3 Summary of problems encountered in the observation period .. 351
N.4 Observation period problems and their resolution ............... 357
N.5 Summary of problems encountered in the observation period .. 357

O.1 Flow of problems into and around COTS: Monday ............. 361
O.2 Flow of problems into and around COTS: Tuesday ............. 362
O.3 Flow of problems into and around COTS: Wednesday ......... 362
O.4 Flow of problems into and around COTS: Thursday .......... 363
O.5 Flow of problems into and around COTS: Friday ............... 363

P.1 Case study Client’s Problems Table ................................. 366
P.2 Case study Client’s Goals Table ...................................... 367
P.3 Case study Client’s Constraints Table ............................... 368

R.1 Case study Stakeholder Stakeholder-one’s Problems Table .... 389
R.2 Case study Stakeholder Stakeholder-one’s Goals Table ........ 390
R.3 Case study Stakeholder Stakeholder-three’s Problems .......... 391
R.4 Case study Stakeholder-three’s Goals Table ...................... 392
R.5 Stakeholder User-One’s Goals Table ................................ 393

S.1 Transforming the client’s problems to goals ....................... 396
S.2 Transforming stakeholder one’s problems to goals .............. 397
S.3 Transforming stakeholder three’s problems to goals ............ 398
S.4 Transforming users’ problems to goals ............................. 399
Chapter 1

Introduction
1.1 Background and motivation

Software engineering concerns the development of computer-based systems (CBSs). Requirements engineering is that part of software engineering concerned with the development of requirements for computer-based systems. One class of computer-based systems comprises applications designed to support organisational activities. This is the class of socio-technical systems. Early approaches to developing requirements for this kind of system were associated with a number of problems (Chapter Two presents their background):

- There is often a neglect of the wider organisational context [McD94, Eas88, Fly98]
- The high-level objectives of an organisation are often ignored [McD94]
- Business needs are often ignored [KS98]
- There is often a poor understanding of the application domain [CKI88, KS98]
- There is often a low involvement of the stakeholders [KS98]
- There is often poor communication among stakeholders [Mac96]
- Stakeholders often resist new systems [Fly98, KS98]
- There is often a misunderstanding users’ work [Mac96, LHG94]
- Often the wrong problem solved [Fly98]
- There are often negative consequences of new systems
- Ineffective systems are often produced [Fly98]

In the last ten to fifteen years a number of approaches have appeared that are designed to tackle these problems. They focus upon the social and organisational side of software-intensive systems development as well as upon the technical, and also tend to be participative in nature, requiring the active involvement of stakeholders. Such approaches include Checkland’s Information System Development (ISD) [WBPB95], Eason’s approach [Eas88], and Contextual Inquiry (CI) [HB93, HB96]. These may all be categorised as socio-technical approaches (see Chapter Three). One of the key features of these approaches is their implicit, or sometimes explicit, use of goal-oriented-like techniques to derive requirements. Actual goal-oriented techniques have been developed explicitly in a number of other recent approaches. The KAOS approach [DvF93] and Loucopoulos’s approach [LK95a, LK95b] are examples of such approaches.

\(^1\)Page 20 and Chapter 4
1.2 Statement of problem

First, although the three socio-technical approaches referred to above do incorporate embryonic forms of the goal-oriented approach to requirements engineering, it is in all cases insufficiently articulated for them to effectively tackle some of the key problems in the list above: high-level objectives and business needs often being ignored, for example. Second, all of the approaches just referred to are characterised by being either poorly elaborated or difficult to apply. For some—Information Systems Development, Contextual Inquiry, and Eason’s socio-technical approach—either the activities involved in an approach are described in insufficient detail, or the syntax and semantics of associated modelling languages is insufficiently clear, or both. For others—KAOS and Loucopoulos’s goal-oriented approach—the approaches are difficult to apply. It is argued here that this low-level of elaboration has contributed to preventing these approaches from being widely taken-up and applied to real-world problems. This, in turn, has meant that it has proved difficult to compare the effectiveness of these approaches with other recent approaches, like the use-case approach [Jac93]. This may mean that a better approach to developing requirements for organisational computer-based systems has been overlooked. And third, some of the problems which the three socio-technical approaches—Information Systems Development, Contextual Inquiry, and Eason’s socio-technical approach—were intended to attenuate or eliminate are still prevalent. In particular, current usable approaches provide little help on selecting which computer-based systems to build out of all those that could be built in a particular context.

1.3 Contribution of thesis

Although building methods is not a fashionable pursuit within the software engineering research community, nevertheless methods remain the core of software engineering practice. A good example of the latter phenomenon is the ubiquitous application of the Rational Unified Process [Kru00] using the Unified Modelling Language [OMG03]. One strategy for building methods is to synthesise a new method from pre-existing methods. And that is the strategy that has been adopted here: a new goal-oriented approach to requirements engineering has been synthesised from a number of pre-existing requirements engineering methods. Substantial validation of the new approach has been achieved by applying it in a large, real-world case study.

In order to select parts of methods from which to synthesise a new approach to address the problems outlined in the previous section, it proved necessary to carry out a detailed investigation, analysis and critical review of the traditional, socio-technical, and goal-oriented approaches to requirements engineering. The investigation and critical review helped to establish a framework of principles and features for guiding the development of requirements engineering methods in general, and the method developed by this thesis in particular. The investigation and review also provided insights into the relationship between the details of the reviewed approaches and the problems associated with them. And these insights helped to generate ideas for the process and models of a new approach, the application of which leads to a significant reduction in some of the problems, and the elimination of
As prescribed by the framework’s principles, the processes and models of the new approach have been described in detail. This should encourage users to apply in order to develop requirements for computer-based systems that support organisational activities. The basic structure of the new approach is presented below:

- Conceptualise the current served system
  - Investigate and model the broad served system
  - Investigate and model external views of the served system
  - Investigate and model the focused served system
  - Summarise investigations and resolve anomalies

- Investigate IT and benchmarks, and select stakeholders
  - Investigate IT (optional)
  - Benchmarking (optional)
  - Assemble stakeholder group

- Derive requirements for new served and serving system
  - Model stakeholders’ goals
  - Identify new served and serving systems (Stakeholder-meeting-1)
  - Identify conflict and outline new served and serving systems (Stakeholder-meeting-2)
  - Resolve goal conflict and refine new served and serving systems (Stakeholder-meeting-3)
  - Document requirements for new served and serving systems

In order to thoroughly assess the effectiveness of the new approach, it was necessary to apply it in a real-world case study, and to review the outcome of this application. The result of this activity indicates that the approach can be applied, that it does identify important computer-based systems for supporting given organisational activities, and that it does help to generate requirements for those computer-based systems.

In summary, the two main results of the research undertaken in support of this thesis are as follows. First, a framework of principles and features for guiding the development of requirements engineering methods has been established. Second, the framework has been used to both guide the development of a new goal-oriented approach to requirements engineering, and to compare it to some established socio-technical approaches, goal-oriented approaches, and process-oriented approaches. The new approach is intended to help practitioners to identify and derive requirements for computer-based systems that support an organisation’s key activities. Because the approach is described in detail, it is hoped that it will be tried by requirements engineering practitioners. To the extent that it is taken up, it will enable goal-oriented approaches to be compared to other approaches to requirements engineering more thoroughly than has proved possible to date.
1.4 Novel features

The new, synthesised goal-oriented approach to requirements engineering is associated with a number of novel features. First, approaches to requirements engineering typically start with a pre-existing, informal statement of requirements and proceed by identifying, expressing, formalising, and organising them. But, in practice, deriving the first statement of requirements is difficult; this is in part because there are no established methods for doing it. The new, synthesised approach provides just such a method for deriving a statement of requirements. In doing this, it adds to the utility of requirements engineering.

Second, although some requirements engineering approaches have considered domain problems as a source of requirements, none have done so to a significant extent. But domain problems are a key indicator of areas where improvements can be made in a domain. The new, synthesised approach emphasises the identification and use of domain problems as key drivers of improvements in a domain; in particular, domain problems are used for determining requirements for associated supporting computer-based systems in the domain.

Third, established goal-oriented approaches to requirements engineering, tend to capture and use just the current goals of a domain for deriving requirements for computer-based systems that will support the domain. But, in doing this, these approaches miss the opportunity to improve the domain by capturing and then supporting new goals. The new, synthesised approach captures both current goals and new goals, and, in doing so, opens up the possibility of enabling more improvement to a domain to be achieved.

Fourth, unlike other prominent goal-oriented approaches, the new, synthesised approach requires the stakeholders themselves to be actively involved in enacting substantial parts of the approach: for example, they are involved in establishing current and new goals for a domain, and in generating incremental and radical ideas for achieving agreed goals. By doing this, the new approach not only avoids the problems associated with not involving stakeholders—rejection of subsequent computer-based systems, for example—but also taps a knowledgeable source of ideas for both goals and the mechanisms to achieve them.

Fifth, unlike other approaches to requirements engineering, the new, synthesised approach encourages stakeholders to fully appreciate the full impact of their ideas for satisfying goals. This is done by getting them to predict additional effects (side effects) of their ideas, and then evaluating those effects—positive, neutral, or negative—with respect to each stakeholder. The results are intended to be used to evaluate alternative suggestions for satisfying goals. This feature of the new, synthesised approach should help to weed out computer-based systems that, while satisfying new, agreed goals, unfortunately produce unwanted side-effects.

Sixth, traditional goal-oriented approaches require a requirements engineer to elicit high-level goals, and the decompose them to create a goal-hierarchy. The hierarchy is used to derive requirements for computer-based systems. This method can lead to the identification and use of lower-level goals that do not correspond to the reality of a domain, and thus to the identification of requirements for inappropriate computer-based systems. By
contrast, the new, synthesised approach elicits goals which may be at a range of levels, from high to low. The approach builds these into a goal-hierarchy, adding “missing” goals if necessary. In doing this, the new approach leads to the identification of requirements for systems that meet goals that correspond to stakeholders’ actual goals.

Seventh, although other approaches to requirements engineering do often feature validation, the new, synthesised goal-oriented approach features it more prominently: stakeholders are required to critically review each important output from the approach. This means that the end result—requirements for computer-based systems—is highly likely to satisfy the client and other stakeholders.

Finally, this work constitutes an attempt to synthesise (and validate) an approach from selected features of three families of approaches to requirements engineering: socio-technical, goal-oriented, and process-oriented. And synthesising an approach from these three particular families of approaches has not been tried before.

1.5 Thesis structure

Following this introductory chapter, Chapter Two first motivates the need for high-quality requirements engineering, then outlines the classical requirements engineering process. Important problems associated with this approach are then listed. In Chapter Three, many of these are seen to be successfully tackled by three socio-technical approaches to requirements engineering: Checkland’s Information Systems Development method [WBPB95], Contextual Inquiry [HB93, HB96], and Eason’s socio-technical approach [Eas88]. Unfortunately, some key problems remain: in particular, applying these approaches is problematic because they are not described in sufficient detail. Chapter Four focuses upon approaches that are explicitly based upon the goal-oriented approach (GOA). First, a rationale and generic description for the goal-oriented approach are presented. Second, a number of established approaches—KAOS [DvF93], Loucopoulos’s approach [LK95a, LK95b]—are considered in detail, and the relationship between goal-oriented approaches and process modelling [Oul95, WKRG99, Dav93] is explored. The chapter includes a summary of the advantages and problems associated with the goal-oriented approach.

Chapter Five analyses the common features of the three socio-technical approaches to requirements engineering, and relates them to the goal-oriented approach to requirements engineering. A generic model for goal-oriented requirements engineering is also presented. This chapter acts as a bridge between the socio-technical and goal-oriented approaches presented in the preceding chapters, and the new, synthesised goal-oriented approach presented in the next chapter. Chapter Six establishes a framework of principles and features, derived from the literature review, for guiding the development of requirements engineering methods. This framework is used to develop a synthesised goal-oriented approach to requirements engineering from the generic model that is presented in Chapter Five. In order to validate the approach, it was applied in a large-scale case study to a University Faculty’s Computer Services Department; the outcome is presented in Chapter Seven. The chapter begins with a justification of the use of the case study method.
for validating the approach.

A critical evaluation of the work undertaken is presented in Chapter Eight. The synthesised approach is assessed against the claims made in the first chapter and its limitations are explored. The thesis is summarised in the last chapter and attention is drawn to its achievements. The chapter concludes by outlining future related work. Such work includes the development of computer-based tools to support the approach—its process activities and models. It also includes the testing of the approach in a wide range of organisational contexts.

1.6 Aids for the reader

Appendix B contains expansions of the acronyms and definitions for some of the technical terms used in the body of the thesis.
Chapter 2

Requirements engineering
2.1 Introduction

This chapter introduces requirements engineering. It first describes the key problems of software-intensive systems engineering—the context in which requirements engineering occurs. It continues by indicating how good requirements engineering may attenuate or eliminate some of these. A traditional requirements engineering process is then described in detail, and this is followed by a review of the main problems associated with methods based upon it.

2.2 Problems of software-intensive systems engineering

Requirements engineering occurs in the context of software-intensive systems engineering. Software-intensive systems engineering is a discipline for developing systems whose main component is software. In order to develop software, the requirements that define its nature must be established first. Requirements engineering is the discipline for establishing requirements for software-intensive systems. It is widely accepted that organisations of all sizes and types experience serious problems developing software-intensive systems. The main problems are listed below:

- Projects are often cancelled
- Projects frequently experience cost overruns
- Projects often deliver systems with less functionality than was agreed
- Projects sometimes deliver systems that do not work
- Projects often deliver systems late
- Poor benefits are sometimes obtained from delivered systems

Donal Flynn [Fly98] has collected evidence from a variety of sources to support the prevalence of these problems. For example, he quotes a report from Computing (24th August 1995) featuring a cost overrun: “The collapse of systems such as the London Stock Exchange’s failed Taurus project . . . has made the financial market nervous about IT . . . By the time Taurus was abandoned in 1993, the cost had ballooned to £100m - twice the original estimates”. And again, he quotes from a UK survey produced by consultants KPMG Peat Marwick McLintock [KPM90] that “30% of the UK’s biggest computer projects were massively over budget, over time, and, if ever completed, failed to do the job they were meant to”. Flynn also presents other supporting evidence based upon academic research. This includes a survey conducted by Lederer and Prasad [LP93] of 115 IT professionals which found that 63% of large projects substantially overspend.

Additional evidence comes from a 1979 US Government survey of nine software development projects that was originally reported in [Dav90]. Davis wrote: “Although the size of the projects was quite small (the sum of the nine contracts was less than $7 million), the findings were depressing: Forty-seven percent of all the dollars were spent for software that was never used!
To make matters even worse, an additional 29 percent of dollars were spent for software that was never even delivered, and another 19 percent resulted in software that was either extensively reworked after delivery or abandoned after delivery but before the GAO (Government Accounting Office) study was conducted”.

An oft-quoted survey of over 8000 projects in over 250 companies carried out by the Standish Group [Gro94] provides further supporting evidence. This survey found that only 16.2% of all projects were completed on time and within budget, with all features and functions as initially specified; that 52.7% of projects were completed and operational but over-budget, over the time estimate, and offering fewer features and functions that originally specified; and that 31.1% of projects were cancelled at some point during the development cycle.

2.3 The solution: requirements engineering

In the past more effort has been expended on the downstream software development activities—design and coding—than on the upstream activities—requirements engineering—in an effort to attenuate or eliminate the problems associated with developing software-intensive systems. However, newer data indicate that the quality of requirements engineering may be a more reliable determinant of project success. For example, when the Standish Group [Gro94] asked IT executive managers for their opinions about why projects succeed, their responses were as follows. Thirteen per cent of responses (the third highest number) indicated that “a clear statement of requirements” was important. The fifth and ninth highest number of responses were for “realistic expectations” and “clear vision and objectives”. So here are three requirements engineering related factors in a list of ten factors that are deemed to be strongly associated with project success. Similarly, “incomplete requirements” and “changing requirements specifications” were seen to be associated with projects that overran budget and time, and failed to deliver all specified functionality. And “incomplete requirements” was perceived to be the most important factor associated with projects that were cancelled. The conclusion here seems to be that IT managers believe the following: if requirements engineering activities are carried out, a project is highly likely to succeed on time and within budget; conversely, if they are not carried out, a project is likely to suffer in some way as a consequence: either cost overruns, budget overruns or incomplete functionality will ensue. In the worst cases the result is project cancellation.

These conclusions accord with similar conclusions derived from a survey of software management practises in Western Europe, Japan and the US undertaken by Blackburn and Scudder [BS96]. In their research, they tried to distinguish management practises that result in faster development speeds and higher productivity. They found that “firms improving their development speed at the fastest rate actually spend more elapsed time and effort in the customer requirements stage of the project—that is in determining what the customer wants in the software before proceeding into high-level planning, designing and coding”. In addition, they found that the firms with the highest productivity also spend more time and effort on user (or customer) requirements: “the faster firms devote an average of 18% of their
development cycle and 14% of their man hours to determining customer requirements. The slower firms expend about one-half the effort in determining customer requirements”.

In addition, Flynn [Fly98] notes that surveys undertaken by de Marco [dM82], quoted in [Fin89], show that most errors in a system are introduced during the requirements phase (56%). The survey also showed that requirements errors were the most expensive kind of error to fix: the 56% of requirements errors required 82% of the error-correction costs to fix. These data also suggest that more resources might profitably be devoted to the requirements phase in order to reduce requirements errors.

So, to summarise, IT managers perceive requirements related factors to be very important for determining the success or failure of a project; fast and high quality development correlate positively with resources devoted to requirements; and devoting more resources to the requirements phase is likely to result in fewer requirements errors and therefore cheaper projects. These findings all emphasise the high value of requirements engineering as part of software-intensive systems engineering.

2.4 Requirements engineering

The previous section describes problems with software-intensive systems engineering and indicates that they might be attenuated if greater attention is paid to requirements engineering. But what precisely is requirements engineering? For example, what activities occur during requirements engineering? And in what order are they performed? At a high level, most requirements for a system to be created express either properties which should hold for the system, or properties which should hold for the process creating the system. Examples of requirements belonging to the first category might include the following: the system should be able to do function x and function y; the system’s user interface should be multi-coloured; the system should be able to complete an update transaction in less than 3 seconds; and so on. Examples of requirements belonging to the second category might include the following: the Jackson Structured Design method should be used to develop the system; the end-date of the project should be 1992; only three grade one engineers should be used on the project; ISO Quality standard 9001 must be adhered to at all times; and so on.

Requirements engineering is a process whose goal is to capture all the requirements (of both the above types) for a system to be created. Requirements engineering consists of the following main stages:

- Requirements acquisition
- Requirements analysis
- Requirements specification construction
- Requirements validation
- Requirements use

It is important to note that this is an over-simplified model. In reality, iteration between stages occurs, and stages are often performed out of
sequence, and more than once. To appreciate the iterative nature of the process, it is worth noting that the work in any stage may require a return to the user, either to obtain new requirements from him or her, or to suggest the modification or removal of existing requirements. To appreciate that stages may be performed out of sequence and more than once, it should be noted that validation may also occur after stage-one, since it is desirable that the users should check even an informal statement of requirements to ensure that it expresses what they want.

Requirements acquisition is normally the first stage in the requirements engineering process. The set of requirements for a system must be determined before the system creation process can start. Thus, the first main function of requirements engineering is to collect all the relevant requirements. This stage of product development is called, variously, requirements capture, requirements elicitation or requirements acquisition. The goal of this stage is to determine the full set of requirements deemed necessary for the development of a system. These are elicited from the various sources of requirements—users, customers, analysts, and so on—and expressed in a suitable form: initially, requirements are usually expressed informally in a natural language like English; later, during the requirement specification construction stage, they may be expressed more formally.

In the second stage, requirements analysis, the main activity is to analyse collected requirements. The goals of this stage are to organise the requirements into logically coherent groups—for example, to collect together all the requirements related to performance—and to critically review them. During the review an attempt is made to determine whether the requirements conflict with each other, whether any requirements are unclear, whether any requirements are stated more than once, and whether requirements for particular areas are missing. On the basis of the analysis, the users and the customers may be confronted by the requirements engineer in order that, together, they may attempt to resolve contradictions, clarify opaque statements, eliminate redundancy, and so on.

During the requirements specification construction stage, the requirements are recorded in a requirements specification document. A variety of templates exist for constructing requirements specification documents. A typical document structure is presented in [Bra02] by Bray.

During the validation stage, the complete set of requirements may be “played back” to the users and customers; they can then determine whether the requirements do represent their actual needs and therefore would, if implemented, result in the product they wanted being created.

2.5 Problems of requirements engineering

Unfortunately, a number of important problems with some requirements engineering methods that are based upon the traditional requirements engineering process just described have been reported by requirements engineering researchers. In an oft-cited study of the software design process for large systems, Curtis et al. [CKI88] found one of the three most salient problems to be “the thin spread of application domain knowledge”. “The deep application-specific knowledge required to build most large, complex systems was thinly spread through many software development staffs”. Sommerville
and Kotonya [KS98] sound the same note when they assert that one of the problems with the effective elicitation of requirements is that “sometimes ... requirements engineers either don’t or can’t learn about the domain”. Poor understanding of an application domain is usually associated with the low involvement of stakeholders—a main source of domain knowledge in a development. It usually also means that higher-level organisational objectives have been ignored or neglected; that the wider organisational, social and political context has been ignored or neglected; and that the users’ work practises have been poorly understood. With regard to the first point, Sommerville and Kotonya [KS98] have noted that “the requirements engineering process does not identify and take into account the real needs of all the stakeholders in a system”.

And again, John McDermid [McD94] has pointed out that specifications produced by traditional requirements engineering methods “may violate the high-level objectives of the organisation the system is intended to serve”. If the high-level objectives have not been previously ascertained, such violations would be hard to spot. McDermid’s view is shared by Sommerville and Kotonya [KS98], who assert that the fact that “business needs are not considered” is another problem with requirements elicitation.

Additionally, a number of writers, for example, [Fly98], [McD94] and [Eas88], have highlighted as problematic the neglect by traditional requirements engineering methods of the wider organisational, social, and political context of software development. For example, McDermid [McD94] declares that “a key problem in many systems is the need to deal with both organisational and technical (i.e. soft and hard) issues, and it seems that there are few, if any, methods that cope well with both sorts of issue”. And again, Flynn [Fly98] highlights one critical organisational factor when he declares that “wider social or psychological factors may be neglected such as ... the extent to which the information system will be acceptable to ... its intended users”.

Also, poor understanding of the work of users has also been highlighted as problematic [Mac96] [LHG94]. For example, Macaulay [Mac96] asserts that “designers do not fully understand the work of users”. And Luff et al. claim that “we don’t watch users either in current work or when they use our systems. So we don’t see their requirements for certain features or certain interfaces” [LHG94].

And all the problems mentioned above may contribute towards creating other problems. For example, the wrong application domain problem may be solved; stakeholders may resist a new system; there may be negative consequences of a new system; and new systems may be ineffective. First, it is not difficult to see that if an application domain is not thoroughly investigated and understood, then it is quite likely that the most important problem in that domain will be missed, and as a consequence any software-intensive system produced will solve the wrong problem. Flynn [Fly98] recognizes this when he asserts that “in many cases the wrong activities to assist are chosen”. Neglect of the wider organisational context and the failure to include a wide range of stakeholders in the development process may lead some stakeholders to reject a new system. This danger is emphasised by Sommerville and Second, Kotonya [KS98] when they say that “in many cases, buying and installing a new system is an organisational
decision and the people who are affected by the system are not consulted. They may feel that a new system is not necessary and they don’t see why they should cooperate in its specification”. And third, Flynn [Fly98] also points out that “information systems are often of poor quality as they do not have a beneficial effect on the organisation”, and also that sometimes “their effect on the organisation . . . at worst . . . may be destructive, affecting employers or customers”.

The list below summarises the main problems associated with some methods of requirements engineering that are based upon the traditional requirements engineering process.

- There is often a neglect of the wider organisational context [McD94, Eas88, Fly98]
- The high-level objectives of an organisation are often ignored [McD94]
- Business needs are often ignored [KS98]
- There is often a poor understanding of the application domain [CKI88, KS98]
- There is often a low involvement of the stakeholders [KS98]
- There is often poor communication among stakeholders [Mac96] 1
- Stakeholders often resist new systems [Fly98, KS98]
- There is often a misunderstanding users’ work [Mac96, LHG94]
- Often the wrong problem solved [Fly98]
- There are often negative consequences of new systems
- Ineffective systems are often produced [Fly98]

### 2.6 Summary

This chapter has described requirements engineering, and placed it in the context of the engineering of software-intensive systems. It has also described important problems with traditional requirements engineering methods. The next chapter describes three socio-technical approaches to requirements engineering; the one after describes the goal-oriented approach to requirements engineering. These approaches are intended to attenuate or eliminate one or more of these problems.

---

1Page 20 and Chapter 4
Chapter 3

Requirements in context
CHAPTER 3. REQUIREMENTS IN CONTEXT

3.1 Introduction

This chapter introduces three socio-technical requirements engineering methods. By focusing upon the social and organisational side of software-intensive systems development as well as upon the technical side, these kinds of method are intended to tackle issues that are ignored by the traditional methods. However, although the three approaches are seen to reduce the problems of the traditional methods discussed in the previous chapter, they are themselves associated with new problems, each of which is presented and discussed. The chapter ends with a discussion of the concept of the term “stakeholder”. This an important feature of the three socio-technical approaches and also of the new synthesised approach (see Chapter Six), but it has been interpreted in a variety of ways in the literature.

3.2 Socio-technical approaches to requirements engineering

The previous chapter reviews some of the main problems that have been encountered with traditional requirements engineering methods: neglect of domain knowledge, neglect of the wider organisational context, neglect of high-level organisational objectives, lack of stakeholder involvement, and others. This section describes three socio-technical methods which each attempt to tackle one or more of these problems. Flynn [Fly98] defines a socio-technical system as “any unit in the organisation composed of a technological and a social sub-system having a common goal or task to accomplish”\(^1\). The social subsystem comprises people interacting with other people and with technology in order to accomplish such tasks. Although other socio-technical methods might have been chosen, in particular, Mumford’s ETHICS method [Mum86, Mum89], the following methods were selected:

- Checkland’s Information Systems Development [Che81]
- Eason’s socio-technical approach [Eas88]
- Holtzblatt’s and Beyer’s Contextual inquiry [HB93, HB96, HB98]

Each method is introduced in turn. First, a description of the method is given. This is followed by a discussion of the extent to which the method should solve problems with requirements engineering that were enumerated in the previous section.

Eason’s socio-technical approach

In his book Information Technology and Organisational Change [Eas88], Eason presents a method for developing software-intensive systems. Eason’s method tackles many of the problems described in the previous section. An outline of his method is shown below. This is followed by a detailed description of the method’s stages.

1. Analyse organisational needs and opportunities

\(^1\)Page 340

27
2. Specify socio-technical options

3. Assess the consequences of each socio-technical option

4. Analyse the user/task requirements of a socio-technical system

5. Specify and build a prototype system

6. Evaluate a prototype system

In Eason’s method the focus of the first three stages is on the organisation as a whole: determining its main goals; generating broad socio-technical systems to satisfy those goals; and evaluating these systems. The final three stages focus on smaller subgroups within an organisation. To quote Eason [Eas88]²: “The boundary for this analysis may be more limited than for the initial socio-technical analysis, for example the transactions performed at the counter by a group of cashiers rather than the entire operation of a branch of a bank”.

During the first stage of the method an attempt is made to understand the underlying nature of a business. This nature is characterised by “the goals being sought and the problems to overcome”³ in the organisation. Eason proposes a top-down approach for establishing the goal hierarchy in an organisation: first ask senior people about their objectives and responsibilities, and then examine how these decompose to give objectives for subordinates. Also in stage-one, an attempt is made to characterise the current socio-technical systems. This includes creating “an outline view of the socio-technical system currently undertaking the work, the social structure responsible for different functions and the way in which the technology is used to support the fulfillment of the functions”⁴. The reasons for making this characterisation are as follows: knowing how an organisation’s work is performed helps one to infer the main goals of that organisation; it also helps one to better understand an organisation’s main problems; it also helps just to understand an organisation, and this is important, because any new system will be introduced into the current organisation.

During stage-two, Eason conceives some stakeholders—selected managers and IT specialists—using the results of the stage-one analysis in order to identify a range of socio-technical options to meet the needs that have been identified. For each socio-technical option, there are “likely to be a variety of technical ways of serving the application, and a number of possible social structures”⁵.

In stage-three the socio-technical options generated in stage-two are evaluated. First, managers assess the extent to which an option will satisfy one or more organisational goals. Second, an attempt is made to assess “the impact of the technical system upon the social system”⁶.

The input to stage-four is an outline socio-technical system. Detail now needs to be added to both the technical and social aspects of this system. This detail comes from a user/task analysis or series of such analyses. It

²Page 97
³Page 87
⁴Page 89
⁵Page 91
⁶Page 92
has already been pointed out that the boundary of each analysis will not be as wide as the organisation, but will encompass some subgroup within the organisation. For each main task within a subgroup, the following process is enacted:

1. Describe a task
2. Identify the task’s inputs
3. Describe the task’s environment
4. Create a process model for the task (actions on objects)
5. Perform rôle analysis and user analysis
6. Analyse existing technical system
7. Specify requirements for new social system
8. Specify requirements for new technical system

The initial steps (one to six, not including four) are concerned with finding out about and documenting aspects of the current system. Steps seven and eight are concerned with specifying the requirements for a new social system and a new technical system respectively. In step-one, a task is described in terms of the hierarchy of goals to be achieved by that task. In step four, an attempt is made to construct a process model which will satisfy those goals. The process model is made up by combining actions on objects in appropriate ways: in sequence, in parallel, and by repeating actions.

Step five comprises a rôle analysis and a user analysis. The former determines how key activities in step four are currently assigned to rôles and individuals within the organisation. This work can be viewed as an extension and elaboration of the attempt in stage-one “to create an outline view of the socio-technical system currently undertaking the work”\(^7\). The user analysis is “an exploration of the characteristics and qualities of the people who will ultimately become users of the new system”\(^8\). It determines their physical characteristics, skills and qualifications, and sources of stress and satisfaction.

Returning to stage-five and stage-six of the main process, in order to evaluate the social and technical requirements of a new system, a prototype is created (stage-five) and evaluated (stage-six). Clearly, before a prototype can be specified, “a statement of how the work performing system undertakes the required work”\(^9\) is needed. This indicates how the technical and social aspects of the new system are combined together.

Having provided a description of Eason’s socio-technical approach to requirements engineering, its likely impact on the problems of approaches to requirements engineering discussed earlier is now assessed. First, Eason’s approach clearly addresses the group of problems associated with a lack of understanding of the broadest sense of the served system. Activities in stage-one of the approach, “Analyse organisational needs and opportunities”, tackle the traditional neglect of the wider organisational context: they

\(^7\)Page 89
\(^8\)Page 102
\(^9\)Page 102
explicitly and prominently attempt to determine the high-level objectives of an organisation, including its business needs. In addition the activities in stage-four, “Analyse the user/task requirements of a socio-technical system”, tackle the problem of poor understanding of the application domain.

Second, Eason’s approach addresses two out of the three problems associated with stakeholders. For example, stakeholders are more involved in this method than in traditional methods: managers are involved in goal elicitation (stage-one); selected managers and IT specialists are involved in specifying socio-technical options (stage-two); each stakeholder group evaluates socio-technical options (stage-three); managers and staff are involved in specifying social system requirements (stage-four); and users evaluate prototype socio-technical systems and provide feedback (stage-six). And again, since there is a stage in which the stakeholder group deemed most likely to suffer from a new system is asked how it would respond to that new system, it seems likely that the incidence of new systems being resisted by users should also be diminished. However, Eason’s approach does not seem to offer explicit support for improving poor communication among stakeholders.

Third, the problem of misunderstanding the users’ work should also be attenuated in Eason’s approach, since, in stage-one, there is an attempt to create an outline of the socio-technical systems currently undertaking an organisation’s work. And in stage-four, there are explicit steps concerned with describing current tasks, how they are currently performed, and how they are currently supported by technology. In addition, it is claimed by Eason that the activities undertaken analysing organisational needs and opportunities should uncover an organisation’s principal problems. To the extent that this is so, one would expect a reduction in the incidence of the wrong problem being solved.

Fourth, Eason’s approach also addresses problems associated with the effects and effectiveness of new computer-based systems. In stage-three, the social and technical consequences of alternative socio-technical systems are predicted and evaluated by stakeholder groups. And in stage-six, prototype socio-technical systems are evaluated by users. Both these activities should help to surface negative consequences of a new system earlier, and thus allow them to be reduced or removed. Similarly, stage-three and stage-six should help to show up ineffective systems, and allow them to be improved before implementation.

Overall, apart from not explicitly improving stakeholder communication, one would expect Eason’s socio-technical approach to have beneficial effects on all the main problems associated with traditional requirements engineering methods. However, there are some problematic aspects associated with this method. These are discussed next.

Although it seems that Eason’s approach would improve requirements engineering by effectively tackling some of its main problems, unfortunately the approach, as presented, itself presents some difficulties. The first major difficulty is caused by the fact that the approach is presented in a relatively informal, broad-brush manner. It would be difficult for someone else to apply this approach as it stands without first having many questions answered. For example, it is not made clear how socio-technical options are to be generated in stage-two using the results of a stage-one organisational needs analysis. Again, during stage-one it is necessary to produce “an outline view of the
current socio-technical system”. But it is not made clear what components comprise such a view, or how each is determined. Nor is it made clear how the top-down approach to determining the main enterprise goals is to be carried out. For example, how are goals to be expressed? How much detail should be included in a goal description? How will large numbers of related goals be managed effectively? Finally, in stage-four, it is not made clear how the results of the analysis of the status quo should be used to derive a new social and a new technical system, nor is it made clear in the next stage how one should go about integrating the two.

The second major problem is that the relatively informal presentation sometimes leads to confusion. For instance, in stage-one, Eason suggests eliciting objectives and responsibilities from senior people. He says that then one should “examine how these decompose to give the objectives and responsibilities of their subordinates”\(^{10}\). However, it is not made clear whether the decompositions are analytically derived from the high-level objectives, or whether they are obtained from eliciting them directly from the subordinates of the senior people.

Finally, although different types of stakeholders are mentioned and shown to be involved in some of the method’s stages, a definitive list of stakeholder types is not given, nor is it made clear how they should be selected. In addition, the different stakeholder types are each used in a rather piecemeal fashion in the approach. This would contrast with a method that chose a group of stakeholders at the beginning of a project, and then integrated the group fully in the development process for the project’s duration.

In conclusion, Eason’s approach contains a number of features that might effectively tackle some major requirements engineering problems. However, it is felt that most of these features need to be more fully elaborated to render this approach fully usable by others in a real-world context.

### Checkland’s Information Systems Development

Over the last thirty years Checkland and his colleagues at Lancaster University have been developing a methodology—the Soft Systems Methodology (SSM)—for tackling “messy” management problems [Che81, CJ90], and a related methodology, Information Systems Development, for developing information systems [WBPB95, CH98]. The latter constitutes a distinct socio-technical approach to deriving information requirements and, therefore, is of interest here. This section introduces Information Systems Development. For details of the Soft Systems Methodology, the reader is referred to Checkland’s book *Systems Thinking, Systems Practice* [Che81], for example.

Early work of Checkland and his colleagues on the Soft Systems Methodology contains short sections on information systems and information system development [Che81, CJ90]. For example, the observation that “systems analysis aiming at information system design . . . must first concentrate on the activity system which the information system is to serve” is found in [Che81]\(^{11}\). And again, Checkland notes that the Soft Systems Methodology (presumably the activities of identifying, defining, and debating relevant activity systems) helps both to focus on the served system, and to identify

---

\(^{10}\)Page 89

\(^{11}\)Page 208
which particular information system should be built out of the many that could be built in a situation [CJ90]\(^\text{12}\). He also notes that the Soft Systems Methodology activity models “can form a cogent basis for information flow models upon which the information design process itself can be based” [CJ90]\(^\text{13}\).

In more recent work, the focus is entirely upon information systems and information system development [WBPB95, CH98]. The key point in this work, as far as Information Systems Development is concerned, is expressed in Checkland’s law of conceptualisation: “a system which serves another cannot be defined and modelled until a definition and model of the system served are available” [Che81]\(^\text{14}\) [WBPB95]\(^\text{15}\). Checkland justifies this law in [CH98]\(^\text{16}\). Here, he says: “this must be so because the nature of the system served . . . will dictate what counts as ‘service’ and hence what functions the system which provides that service must contain”. This law clearly implies that analysis of the organisation, or that part of it that includes the problem situation, must precede the development of any information system that is to support it. However, for Checkland, this conclusion presents difficulties since the concept of organisation is itself viewed by him as problematic.

In his recent book, Information, Systems, and Information Systems [CH98]\(^\text{17}\), Checkland points out that most of the literature of both the information systems and organisation theory fields contain just one view of the concept of organisation. However, he also notes that some of the literature of both fields, and in particular that of organisation theory, discuss challenges to the prevailing concept of organisation from a number of other conceptualisations of organisation. Clearly, the existence of a range of conceptualisations of organisations is important, because each one has different implications for information system support [CH98]\(^\text{18}\).

Checkland characterises the conventional view of organisations as follows. “Organisations are social units whose members collectively pursue declared objectives . . . the basic organisational process is seen as rational decision making in pursuit of the explicit aims . . . the rôle of an information system is to provide the information which supports the decision-making at various levels from the strategic to the operational” [CH98]\(^\text{19}\). The key points here are that the organisation is taken to have declared objectives, presumably produced by leaders of the organisation—CEO or Chairman of the Board, for example; these objectives are known to, and shared by, the organisation’s members; and the basic activity is one of rational decision-making by members at all levels of the organisation, in order to optimise the achievement of the objectives.

Checkland is not sympathetic to the conventional conceptualisation of organisation. After discussing a number of alternative views and noting that there is no dominant alternative model, he proposes the following characterisation of organisation. Organisations are “cultural processes in which social

---

\(^\text{12}\)Page 53
\(^\text{13}\)Page 53
\(^\text{14}\)Page 237
\(^\text{15}\)Page 134
\(^\text{16}\)Page 111
\(^\text{17}\)Chapter Three
\(^\text{18}\)Page 78
\(^\text{19}\)Page 85
reality is continually defined and redefined in both the talk and action which carries and expresses the multiple agendas of both organisations’ members and significant non-members outside the organisation” [CH98]\textsuperscript{20}. The key point here with regard to Information Systems Development is that organisation members are not assumed to share a single set of publicly declared organisational goals. Some members and groups of members are assumed to have conceptualised the purposes of the organisation in ways other than, and possibly conflicting with, the publicly declared purposes. It is because of this that a variety of systems may be perceived as relevant to participants in a problem situation. Another consequence is the necessity to strive for accommodation among participants with different objectives in any attempt to improve the problem situation.

After discussing alternative conceptualisations of organisation, he goes on to consider the processes which information systems support. “Given three things: the broad concept of what we mean by an organisation . . . the idea of ‘information’ as selected data . . . to which meaning has been attributed in a particular context; and the basic premise . . . that information systems exist to serve and support people taking purposeful action, we can now begin to enrich the concept of . . . an information system” [CH98]\textsuperscript{21}. Checkland does this by exploring the nature of different processes that go on in an organisation. He describes the “personal process”, the “social process”, and the “organisational process”. The latter is a development of the first two and the reader is referred to [CH98] for further details of these.

Checkland says [CH98]\textsuperscript{22} that “the [organisational] process will be one in which the data-rich world outside is perceived selectively by individuals and by groups of individuals”. “Perceptions will be exchanged, shared, challenged, argued over, in a discourse which will consist of the intersubjective creation of data and meanings. Those meanings will create information and knowledge which will lead to accommodations being made, intentions being formed and purposeful action undertaken”. “Adjunct to this process will be another in which the information system needed to support the action will be defined and realised using, usually, appropriate IT and telecommunications, the rôle of IT-based information systems being to serve and support people taking purposeful action in their situations” [CH98]\textsuperscript{23}. From this understanding of organisations and the organisation process, Information Systems Development can be seen as comprising two basic phases: organisation analysis and information systems engineering. This is represented in figure 3.1 as a process using Ould’s Role Activity Diagram (RAD) notation [Oul95]. Appendix C provides a key to this notation, but, briefly, the figure depicts a process comprising one role, the Requirements Engineer; this role is associated with two activities, which are carried out in sequence, with “Perform organisational analysis” occurring first. The organisational analysis that is undertaken in a problem situation in order to determine information requirements for a data manipulation system comprises the following main stages [WBPB95]\textsuperscript{24} [CH98]\textsuperscript{25}.

\begin{itemize}
\item [\textsuperscript{20}]Page 85
\item [\textsuperscript{21}]Page 98
\item [\textsuperscript{22}]Page 104
\item [\textsuperscript{23}]Page 105
\item [\textsuperscript{24}]Page 135
\item [\textsuperscript{25}]Page 112
\end{itemize}
1. Identify and define relevant systems
2. Create corresponding human activity system models
3. Compare the models with reality and debate the results
4. Produce accommodations modelled as human activity system models
5. Analyse the models for information requirements (operational and performance)

In more detail, stage-one establishes “the views of the organisation and its purposes which are relevant and meaningful to the people in the organisation” [WBPB95]26. In stage-two, it is necessary “to conceptualize the activities necessary to pursue these purposes”. During stage-three, one needs to “debate these (served system) activities with people in the organisation, comparing them with current action”. “The required outcome of this debate is a set of meaningful conceptualizations of action relevant to the organisation, to which information analysis can be applied”. Finally, in stage-five, the activities in the HAS models are examined to help to derive the information requirements of anyone carrying out the activities. Support could take the form of either automating action currently performed by people, or of providing informational support to people as they carry out their tasks [CH98]27. Information support might be either to directly support the performance of activities (operational requirements), or to support the monitoring and control of activity performance (performance requirements).

Having provided a description of Checkland’s socio-technical approach to requirements engineering, its likely impact on the problems of traditional approaches to requirements engineering is now assessed. First, Checkland’s
CHAPTER 3. REQUIREMENTS IN CONTEXT

Information Systems Development approach does address the group of problems associated with a lack of understanding of the broadest sense of the served system. Checkland’s Information Systems Development methodology features an organisational analysis as its first main stage. The purpose of the analysis is to understand the organisation, and, in particular, that part of it deemed problematic. The understanding is in terms of the range of purposeful behaviours (in other words, systems) perceived as relevant by the participants. Such organisational analysis should lead to good domain understanding. However, for Checkland, the notion of objectives and goals, associated as it is with the conventional unitary goal-seeking concept of organisation, is itself problematic. Nevertheless, it is certainly not the case that high-level objectives are ignored. In fact, the methodology prescribes that attempts be made to determine the possibly different, and even mutually conflicting, objectives of individuals and groups within an organisation. This occurs in its early stages, when the systems perceived as relevant to the participants are named and defined.

Second, Checkland’s approach clearly addresses the problems associated with stakeholders. For example, with regard to the involvement of stakeholders, Stakeholders are involved in the Information Systems Development methodology at two main stages. They are first involved during the initial stages when the analysts are investigating the problem situation. At this time, the would-be problem solvers interact with the stakeholders (in other words participants in a problem situation) in order to learn about the problem situation. No techniques are formally prescribed; the choice is left to the analysts’ judgement. However, structured and unstructured interviews and questionnaires as well as workshops have all featured in case studies reported by Checkland and his colleagues [CJ90]. Stakeholders probably make their most significant contribution in stage-three of the Information Systems Development methodology. During this stage, activity models based on relevant systems are compared with reality by the analysts and stakeholders. The results of the comparison are intended to trigger a debate among the stakeholders that is intended to lead to accommodations (hopefully constituting improvements) among the stakeholders’ varied interests. In addition, in stage-three, Checkland’s Information Systems Development methodology also promotes stakeholder communication. It does so to the extent that the stakeholders debate the comparison of the human activity systems with reality. Debate implies the idea of people assembled together expressing and hearing (and hopefully attending to) one anothers’ views; debate thus promotes communication. And also, stakeholder resistance to new systems is likely to be relatively low. First, stakeholders are included at significant points in the determination of requirements for such systems. And, second, the improved systems are based upon accommodations achieved by the stakeholders and analysts for the benefit of the stakeholders.

Third, with regard to the problem of misunderstanding users’ work, misunderstanding is always possible when humans try to understand a situation. However, both the initial investigation of an organisation, and the comparison of the human activity system models with reality should facilitate understanding. This would include understanding of the current work of direct users of existing computer-based systems. In addition, comparison of relevant systems with reality and the subsequent structured debate
lead ideally to accommodations among stakeholders over desired improvements. Since the problem situation is thoroughly understood, and since the stakeholders agree on the improvements, it seems unlikely that the wrong problem would be solved.

Fourth, Checkland’s approach does not explicitly address the problem of the unwanted side-effects associated with new computer-based systems, and it is not clear whether or not it addresses the problem of the effectiveness of new computer-based systems. For the former problem, in the methodology there does not seem to be a stage during which the stakeholders or the analysts formally explore the predicted consequences of a solution system (that is the accommodations deemed to constitute a solution). If this does happen informally, one would expect it to occur during stage-three. If this does not happen then the problem of negative consequences is likely to continue to occur. For the latter problem, in some of Checkland’s work, for example [Che81, CJ90, CH98], the form of an improvement to a problem situation varies, and is vaguely defined. For example, Checkland says in [Che81] that “Changes of three kinds are possible: changes in structure, in procedures, in attitudes”. Again, Checkland [CH98] states “it is the debate . . . which would enable actions to be taken in the situation”. However, in [WBPB95], the outcome of the debate is characterised specifically as one or more human activity system models: “The required outcome of this debate is a set of meaningful conceptualisations of action relevant to the organisation, to which information analysis can be applied. More specifically these conceptualisations can take the form of human activity system models whose information support people would like to see exist in the organisation”. It is difficult to find any reference to the analysis for effectiveness of such models produced at this stage. However, [CJ90] points out that all HAS models include a monitoring and control subsystem. It also seems to imply that the model builders must determine, at the time of building, criteria for the efficacy, efficiency and effectiveness of a model, to be used by the monitoring and control subsystem. This seems to imply that the model builder will have accepted at first sight that the activities constituting the model are at least efficacious (and efficacious means to Checkland what effectiveness means here). To the extent that this kind of evaluation is performed in stage-three, the effectiveness of systems developed from such HAS models is likely to be high.

The review above shows that Checkland’s socio-technical approach to requirements engineering addresses some of the problems associated with traditional approaches to requirements engineering. However, the approach is also associated with potential problems. First, both the Soft Systems Methodology and Information Systems Development methodology are methodologies rather than methods. As such they are intended to supply guiding principles to analysts. Such principles are intended to be flexibly interpreted in each problem context to inform the analysts’ activities. As methodologies, they provide neither a definitive list of activities to be performed in each and every situation, nor a comprehensive set of techniques and well-
defined modelling notations to support such performance. This absence of
detailed prescriptive guidance and of supporting techniques and modelling
notations may lead to problems. In particular, newcomers to the method-
ologies may find them difficult to apply, since newcomers usually require
the kind of support that detailed prescriptions offer. Unless they were very
bright and resourceful, some may either use the methodologies incorrectly,
or they might come to a stage in a methodology where they do not know
with confidence how to proceed.

As an example of this phenomenon, the methodologies seem to
offer little in the way of principles, let alone detailed advice, for conducting
a structured debate among stakeholders. It is well known that such debates
are beset by a number of common problems [Mac94]. For example, when
the stakeholders are of different status, the lower status individuals may be
politically unwilling or psychologically unable to contribute. Again, it often
happens that one dominant individual may hog the talk-time, and also force
his or her view upon the others. But Checkland not only fails to refer to
such problems, he also refrains from offering a detailed description of pos-
sible structures and processes that might be used to embody such debates.
Jackson [Jac91] elaborates this point. He says that in order to justify the
changes in a social situation that use of Soft Systems Methodology or Infor-
mation Systems Development might bring about, full and effective partici-
ipation of all stakeholders is required in free and unconstrained discussions.
Only under these conditions can an authentic consensus or accommodation
over changes be reached by the stakeholders. But in many organisations
and social contexts these conditions do not hold. Instead, there is an un-
even distribution of power, status, and access to information resources that
tends to lead both to a lack of commitment by stakeholders to participation
in debates, and to a framework for debates constrained in favour of the most
powerful. And, since Soft Systems Methodology offers little help to redress
such inequalities, it is not likely to produce radical and universally accepted
change. In fact it is more likely to reproduce the status quo.

Another example of the lack of detail concerns human activity system
models when they are used to express improvements to a problem situation.
Human activity system models are constituted only from descriptions of ac-
tivities expressed using English verbs. But, such models still leave open for
would-be problem-owners to solve the question of how such activities are
to be carried out: for example, what different organisation structures and rôle
combinations may be used to “implement” a Human Activity System
model? Surely such information would be much more useful to the would-be
problem solvers than the activity diagrams alone. Related to this point is
the observation that there seems to be no formal part of the final stages of
the methodology concerned with determining all the probable effects of an
“improvement”, and with finding out how the different stakeholders might
variously evaluate each effect. Clearly, such information could help to choose
between different implementations of an agreed “improvement” (human ac-
tivity system model model), and then help to select an improvement imple-
mentation likely to be acceptable to most, or all, stakeholders.

In conclusion, Checkland’s Information Systems Development Method-
ology is based upon the principle that the served system should be concep-
tualised before the serving system. It thus comprises two main stages: first, organisation analysis is carried out to determine information requirements; then, information system engineering is performed to build the corresponding data manipulation system. Because of this emphasis on organisational analysis as the first activity, many of the identified problems of requirements engineering are likely to be attenuated in practice, for example: poor domain understanding, wrong problem solved, misunderstanding users' work, and others.

However, both Checkland’s approach to tackling difficult problems in general (SSM), and his approach to developing information systems in particular (ISD) are methodologies—principles of methods—not methods. Because of this, much of the detailed prescriptive advice usually supplied by traditional Information Systems Development methods, as well as the existence of well-defined procedures, techniques, and supporting notations, is lacking. As a consequence, its seems likely that his Information Systems Development methodology may prove very difficult for some, particularly novices, to use correctly.

### Contextual Inquiry

Ethnographic approaches to requirements engineering have been proposed, for example see [SBRS94]. One of the key problems with using these approaches to develop computer-based systems in the real world is the length of time—months or years—required to carry out the observational phase. Contextual Inquiry (CI) [HB93, HB96] is a method for developing computer-based systems that has been adapted from ethnographic approaches so that the time taken for the observational phase is much reduced: hours or days rather than months or years.

A practitioner following the Contextual Inquiry method first tries to understand the users’ current work, and then uses this understanding to inform the design both of new work practices, and also of any computer-based systems required to support them. Thus Contextual Inquiry is conceptually similar to Business Process Re-engineering (BPR) [HC93], particularly the variant championed by Davenport [Dav93]. Contextual Inquiry comprises the following main stages:

1. Design-team investigate users’ current work
2. Design-team model users’ current work
3. Design-team redesign users’ work
4. Design-team design computer-based systems to support redesigned work
5. Design-team and users refine the computer-based systems designs

During stage-one of Contextual Inquiry, the design-team investigate the users’ current work in order to understand it. Holtzblatt and Beyer state that “Design teams need extensive, detailed information about customers and how they work to build systems that support them well” [HB93] 33.
CHAPTER 3. REQUIREMENTS IN CONTEXT

To obtain this understanding, the design-team observe a range of users at work at the customer site; each design team member observing closely one user at a time. During the observation they ask the users questions both to clarify their understanding of the work being performed, and to elicit the implicit knowledge users employ to work competently. For example, a user might be asked: “What are you doing now? Isn’t there a policy for this? Is that what you expected to happen?” [HB93]. To complement the understanding of the users’ work gained on, and therefore perhaps specific to, the day of observation, Holtzblatt and Beyer claim that “talking about work as it happens, artifacts created previously, and specific past projects reveals the users’ job beyond the work done on that day” [HB93].

During stage-two, the whole design-team share knowledge of the users’ current work practices, through analysing and modelling the work practices of individual users. In a room set aside for the exclusive use of the design-team, the interview transcripts are analysed for key features of the users’ work. These are first written on post-it notes and stuck on to a wall. Then the notes are grouped together into related sets of features. Finally, these groups are named and collected into higher-level groupings. The resulting structure is called an Affinity Diagram (AD). When it is complete, the designers walk through the Affinity Diagram saying what each part is about and brainstorming design ideas for that part. These design ideas are attached to the Affinity Diagram.

The shared knowledge of the users’ current work practices is used to create five kinds of model to represent important aspects of the users’ current work: a work context model, a physical environment model, a flow model, an activity sequence model, and an artifact model. Context models show how organisational culture, policies, and procedures constrain and create expectations about how people work and what they produce. Physical models represent the relevant aspects of the physical environment for the users’ work. “They show how work is split across sites, buildings, and rooms, and how these spaces relate to each other. They show the organisation of work places—the tools, artifacts, and work areas, and their relationship to each other . . . and they show hardware and software where it is used” [HB96]. Flow models depict the rôles—sets of responsibilities and associated tasks—that are associated with the work. “Flow models also show the communication and coordination between rôles and the flow of artifacts between rôles” [HB93]. The flow model “identifies rôles that are necessary to get the actual job done, and which could be eliminated (which normally means removing or automating the tedious parts of a person’s job, not the entire job). The flow model determines the communication paths and artifacts the system should support or replace” [HB96]. Sequence models represent sequences of actions for important work activities. For example, they might be used to depict the coordinated activities of multiple individuals, the steps to take in using a tool, or the thought steps comprising one activity. “They show the intent, or motive, for the activity and for sets of steps, and they show the trigger (the event that caused the sequence to be
CHAPTER 3. REQUIREMENTS IN CONTEXT

initiated)” [HB96]. Finally “artifact models reveal the detailed structure of artifacts created and used to support the work” [HB96].

In stage-three, the models produced in stage-two are used by the design team to redesign the users’ work practices. The main steps of stage-three undertaken by the design-team are shown below:

1. Consolidate work models
2. Walk through the Affinity Diagram and Consolidated Work Models (CWM)
3. Brainstorm alternative Consolidated Work Models
4. Merge the best parts of all Consolidated Work Models
5. Create scripts to show new work practices

Creating Consolidated Work Models involves removing the most customer or user specific detail in order to create a model that might be relevant to a large number of customers. Once produced, the Consolidated Work Models and Affinity Diagram are analysed for inherent problems and inefficiencies. At the same time, users are investigated to see if any have identified ways of fixing problems or attenuating inefficiencies. Also, the members of the design-team brainstorm alternative work models. The best parts of all these new ideas are incorporated into scripts that express new work practices. “Each consolidated sequence (model) drives a redesigned script that shows how that task will be done in the new system. Each step in the consolidated sequence must be accounted for in the redesign; either the step must be supported, the intent of the step must be met in some new way, or the step is obviated because changes to a higher level intent made this step unnecessary” [HB96]. Scripts show the rôle performing a work step, a work step, the changes to the physical environment, and artifacts and concepts used in the step.

In stage four, attention turns to designing computer-based systems to support them. This design-team activity involves two main steps for each computer-based system required:

1. Describe the functionality of the computer-based system functionality (User Environment Design)
2. Create a user interface for the User Environment Design

Holtzblatt and Beyer feel that many methods home in too soon on designing the user interface. For them it is more important to first determine what functions are required and how they are organised (User Environment Design) before determining how the functionality will be represented to users (user interface). They say [HB93] that “our initial and primary concern in system design is to ensure that the structure of the system we deliver supports the work as we redesigned it. We want to be sure that the

39Page 314
40Page 317
41Page 325
42Page 99
system delivers the right functions and organises them to let people work efficiently”. In Contextual Inquiry the organisation of functions to be provided by a computer-based system supporting new work practices is expressed in a model called a User Environment Design. According to [HB96]43: “Redesigned scripts drive the development of User Environment design. The team walks each script extracting implications for the system design. They define places in the system to support each work activity, and define system function, work objects, and flow between places.” Once the User Environment Design has been designed, the design-team design a User Interface for it. The “primary concern in designing the UI is having it realise the UED and [to] make working within the system convenient” [HB93]44.

In the final stage, the cross-functional team of designers and users together refine the design of both the new work and of the computer-based systems that support it. They do this by having the users try out a range of increasingly sophisticated prototypes in their work environment, starting with paper-based ones. When users find a problem either with the new work, or with a computer-based system’s functionality, or with a user interface, they are encouraged to work with the designers towards producing solutions for the problem. In this way the the users and designers can be viewed as working together to refine the system design created by the designers. It is accepted at this stage that working once more with the users means that new learning may take place about the users’ work. This may lead in turn to iteration through the whole sequence of the stages of Contextual Inquiry.

Having provided a description of Contextual Inquiry, Holtzblatt’s and Beyer’s socio-technical approach to requirements engineering, its likely impact on the problems associated with traditional approaches to requirements engineering is now assessed. First, Contextual Inquiry does not address very well the group of problems associated with a lack of understanding of the broadest sense of the served system. The method’s Context Models are supposed to express the constraining of such wider domain features as organisational culture, policies, and procedures. However, the literature does not indicate how these features are to be captured, and it seems unlikely that complete and accurate information on these important domain characteristics could be obtained from the users alone. In particular, the Contextual Inquiry method does not concern itself explicitly with high-level objectives or with business needs. However, since the Contextual Inquiry method advocates the investigation of users’ current work, one would expect it to produce a very a good understanding of the application domain: the users and their immediate environment.

Second, Contextual Inquiry does address problems associated with stakeholders. For example, with respect to the involvement of stakeholders, the Contextual Inquiry method requires a relatively high involvement by a wide range of stakeholders. First, the key design activities—producing both new work practices and the computer-based systems to support them—are performed by a cross-functional design team comprising a wide range of stakeholder types. Second, the users themselves are passively involved in the initial phases of Contextual Inquiry. And, third, these same users are more actively involved in the final stages, when, together with members of the

---

43Page 328
44Page 101
design team, they help to refine the design of supporting computer-based systems. As another example, although Contextual Inquiry does not provide explicit support for facilitating communication among stakeholders, nevertheless the relatively high involvement of stakeholders is likely to lead to informal communication among them. Similarly, this high involvement is likely to lead to greater acceptance of new computer-based systems.

Third, the Contextual Inquiry method is predicated upon understanding the users’ work. So practitioners following this method should acquire a very good understanding of the users’ work. And although the Contextual Inquiry method focuses primarily upon the users’ work, it does use a context model expressing wider domain features, and it also uses a cross-functional design team. The context model includes the representation of organisational structure and the constraints exerted by organisational groups upon each other, although Contextual Inquiry does not indicate how these are determined. The cross-functional design team includes a range of stakeholders. Both of these Contextual Inquiry features should facilitate the introduction of knowledge of the wider domain context, including its problems, into the design situation. And, since problems of the wider context are usually more important than those of the narrower context, one would expect Contextual Inquiry to help in solving the “right” problem.

Fourth, Contextual Inquiry does address the problems associated with the effects and effectiveness of new computer-based systems. Contextual Inquiry requires that designs are embodied in a range of increasingly sophisticated prototypes, each of which is presented, in turn, to the users for them to validate. Thus, the users are given the opportunity to discover effects of a potential new system that they might characterise as negative. They are also able to influence the presence of such effects in the final system. Because of this, one would expect that the negative consequences of a new system would be relatively low, at least as far as the users are concerned. However, this opportunity to validate a new system is not given to other stakeholders types, for example users’ managers, system administrators, and so on. So there may be negative consequences for these stakeholders that are only discovered when a system goes into production. And for the same reasons as for “Negative consequences of a new system” it seems likely that computer-based systems produced by the Contextual Inquiry method will be effective from the users’ point of view, but not necessarily from other stakeholders’ point of view.

The review above shows that Contextual Inquiry addresses some of the problems associated with traditional approaches to requirements engineering. Unfortunately, there are some potential problems intrinsic to Contextual Inquiry:

- The Contextual Inquiry method focuses primarily upon the work of the immediate users. But knowledge of the wider context is also required for context models, and it is not clear whence such knowledge is obtained. Presumably some could be supplied by members of the cross-functional design team, but this is not made explicit.

- The Contextual Inquiry method does not acknowledge that different users, let alone different stakeholder types, may have different ideas about their work and how it is performed. To the extent that such
differences do exist in any real situation, conflict is likely; but the Contextual Inquiry method offers no guidance on how to either record or resolve such conflict.

- Users do not seem to be given the opportunity to validate new work practices separately from any computer-based systems intended to support them. But these are separate features and should be validated separately. First, the new work practices should be validated, then computer-based systems intended to support them should be validated.

- It is not clear what the symbols in a context model—arrows, dots, dashes and circles—mean separately, nor what they mean in combination. Also, it is not clear what kind of text is either allowed or required in a context model. In [HB96]\(^{45}\), it is claimed that “the context model drives design by showing what changes in culture customers are willing to accept”. But it is not clear how this diagram [HB96]\(^{46}\) shows such changes. The text in the diagram refers to job descriptions, agencies, problems perceived from one point of view, requirements, and constraints. However, it does not seem to refer directly to changes in culture acceptable to the customer.

This lack of clarity over meaning also extends to the other types of diagram.

- It is not clear from the two papers whether affinity diagrams are created before Work Models, or vice versa, or whether the two are created in parallel.

- When the designers walk through the affinity diagram, it is not clear whether they are producing design ideas for computer-based systems to support current work, design ideas for new work practices, or design ideas for new work practices and the computer-based systems to support them.

As well as these problems, Contextual Inquiry is also associated with particular strengths:

- Before designing new work practices, let alone any computer-based system to support them, the Contextual Inquiry method requires that the users’ current work be investigated and understood. In this Contextual Inquiry, can be viewed as exemplifying Checkland’s principle of conceptualising the served system before the serving system (see section 3.2).

- Knowledge of users work practices—current and new—is expressed in different kinds of model, each one focusing on a different, but important, aspect of work.

- The design of new work practices is based upon a critical analysis of existing work practices.

\(^{45}\)Page 312
\(^{46}\)Page 312
• Only after the new work practices have been designed does attention in the Contextual Inquiry method turn to designing computer-based systems to support them.

• The design of computer-based systems to support the new work practices is driven by those new work practices.

• The derivation and organisation of functionality required in a computer-based system (User Environment Design) is designed before its user interface.

• The sequence work model models not only networks of activities, but also, importantly, the intents or motives of those activities, and the triggers that cause them to be enacted. The terms “intents” and “motives” are viewed as synonyms for the term “goal”, which is the key concept of this thesis.

• Design of new work practices is, in part, driven by a consideration of the intents and motives (goals) of activities of the current work. The principle of goals driving design is the key one of this thesis, although, in the thesis, the design is driven by existing and new goals, rather than just by existing goals.

• The Contextual Inquiry method recommends the use of a distinct physical space, usually a whole wall, for developing distinct models. For example, one wall might be used for modelling the users’ current work, and another for the new work practices. Clearly, this helps developers to keep separate in their minds aspects that should be kept separate.

In conclusion, the Contextual Inquiry method derives requirements for computer-based systems as follows: first, investigate and critically analyse users’ current work; then, use the results of this analysis to design new work practices; finally, design computer-based systems that will support the new practices—functionality first, then user interfaces. Contextual Inquiry uses an adaptation of ethnographic approaches to investigate the users’ current work. It also employs a form of Business Process Re-engineering, featuring goal-driven design, to design the new work practices and supporting computer-based systems.

The key criticisms of Contextual Inquiry are as follows. It does not seem to pay sufficient attention to the wider organisational context of the users’ work, nor does it explicitly recognise and manage serious differences of opinion that may arise among users over their work—its nature, motives, and activities. As with the other methods and methodologies examined so far, the meaning of diagrammatic notation is not always clear. And, finally, although the users are able to validate the computer-based systems designed to support the new work practices, other stakeholders are not; and even the users do not seem to be given the opportunity to validate the new work practices before validating the computer-based systems intended to support them.
3.3 Problems of the socio-technical approaches

Despite all of the above, the three socio-technical approaches are themselves associated with other problems. The two main ones are that descriptions of the approaches lack detail, and notations for models lack adequate definitions of syntax and semantics. Both problems would tend to make it difficult for people other than the designers of these approaches to use them. All the problems are listed below and then described in more detail:

- Descriptions of the approaches are, in general, insufficiently detailed
- Descriptions of the approaches are sometimes incomplete
- Descriptions of the approaches are sometimes unclear
- Descriptions of models in some of the approaches have unclear syntax and semantics

In general, none of the three approaches is sufficiently detailed for it to be picked up and used without difficulty. It is often not clear how a particular approach activity should be performed, nor what kinds of information should be either considered by, or produced by, the activity. For example, in Eason’s socio-technical approach, it is not clear how the top-down approach to determining the main enterprise goals is to be performed. Nor is it clear either how much, or what kind of, detail should be included in a goal description. For an example from Contextual Inquiry, it is not clear how knowledge is acquired about the context in which the immediate users are embedded. And, for an example from Checkland’s Information Systems Development approach, it is not clear how debate among stakeholders is to be facilitated.

The descriptions of the approaches seem to be incomplete in at least three ways. First, some do not support well known requirements engineering phenomena. For example, the problem of inconsistent requirements is well known within the requirements engineering community, but the Contextual Inquiry approach does not seem to support the management of inconsistent descriptions given by different users of the same work, or the management of different solutions given to the same work problem. Second, some do not always support accepted software engineering practice. For example, ensuring that users validate artefacts produced by software engineers in general, and requirements engineers in particular, is accepted software engineering practice. But Contextual Inquiry does not require the users to validate the new work practices before computer-based systems are designed to support them. And third, some are incomplete in the sense that they do not include simple, but useful, extensions. For example, in Information System Development, a human activity system model could be usefully used to derive a variety of organisation role/structure configurations that satisfy it. And the same approach could easily be extended to include a step that explicitly tries to predict and evaluate the effects of any proposed “improvement”.

Some parts of the descriptions of the three approaches are unclear. Other parts are confusing. As an example of the latter consider the Contextual Inquiry approach. Here, it is not clear whether affinity diagrams are created before work models, after work models, or in parallel with work models. As
an example of the former, consider stage-one of Eason’s approach. Here, it is not clear whether low-level objectives should be analytically derived from high-level objectives, or elicited from the subordinate roles that are responsible for achieving them.

Finally, none of the approaches used notations with clearly defined syntax and semantics with which to construct relevant models. For example, in Contextual Inquiry’s context diagram, the meaning of the symbols is not made clear; nor is it easy to see what various combinations of symbols mean.

3.4 Stakeholders

The term “stakeholder” has been used frequently in the descriptions of the three socio-technical approaches presented in this chapter. Up until now it has been assumed that individual readers’ intuitive understanding of the term would be sufficient to enable them to understand the descriptions. However, in reality the term has been defined and used in different ways by different groups. In addition, because this thesis is concerned with creating a detailed approach to requirements engineering that may be easily used by software developers, it needs to present a detailed description of how to identify and make use of stakeholders. For both of these reasons, this section presents a review of the concept of stakeholder.

A recent review [Pou99] of the concept of stakeholder as it is expressed in the strategic management literature and the information systems literature summarises its use by reference to a framework suggested by Donaldson and Preston [DP95]. This framework characterises stakeholder theory along three dimensions—descriptive, instrumental, and normative—as follows [Pou99]:

- “Stakeholder theory is descriptive in the sense that ‘it describes the corporation as a constellation of cooperative and competitive interests possessing intrinsic value’”.
- “Stakeholder theory is instrumental ‘because it establishes a framework for examining the connections, if any, between the practice of stakeholder management and the achievement of various corporate performance goals’”.
- “Finally, the fundamental basis of stakeholder theory is normative and involves acceptance of the following ideas: ‘stakeholders are persons or groups with legitimate interests in procedural and/or substantive aspects of corporate activity’ and ‘the interests of all stakeholders are of intrinsic value’”.

The review observes that most users of the stakeholder concept in information systems do not “go beyond a descriptive or instrumental application of the concept”. As an example of the descriptive application, consider the work of Boddy et al. [BB86]. For Boddy, “organisations can be viewed as comprising different ‘stakeholders’ groups whose interest in promoting or resisting change, or apathy to innovation, maybe explained by identifying their respective perceived interests ...” Perceived interests here refer to, for instance, job security, promotion prospects, and status. Different stakeholders may view a new technological change—for example the instigation
of a new information system—in different ways depending upon how they perceive it will affect their interests. And this will lead to either cooperation with, or resistance to, the development and introduction of the change.

Of more interest to this thesis is the instrumental dimension, the dimension concerned with the link between stakeholder management and organisation goals. Here “the information systems literature thus concentrates on how stakeholder analysis . . . can support the successful development or implementation of information systems” [Pou99]. In this context, Macaulay, in her book Requirements Engineering [Mac96], first takes the concept of stakeholder as defined by Mitroff [Mit80]: “stakeholders are defined . . . as all those who have a stake in the changes being considered, those who stand to gain from it, and those who stand to lose.” She then proposes that stakeholders for computer systems fall into four distinct categories:

- “Those who are responsible for its design and development: for example, the project manager, software designers, communications experts, and technical authors.”
- “Those with a financial interest, responsible for its sale or for its purchase: for example, the business analyst, the marketing manager, or the buyer.”
- “Those responsible for its introduction and maintenance within an organisation: for example, training and user support staff, installation and maintenance engineers, and user managers.”
- “Those who have an interest in its use: for example, user managers and all classes of users, primary, secondary, and tertiary.”

The primary, secondary, and tertiary users in the last category refer to the scheme proposed by Eason [Eas88]: primary users are the main predicted users of a computer-based system; secondary users are the predicted occasional users; and tertiary users are those who are likely to be affected by a new system, but are not direct users of it.

Macaulay’s categorisation of stakeholders, incorporating Eason’s classification of user types, provides a clear and simple framework for analysing stakeholders in the context of computer-based systems development. And because it is clear and simple, it has been chosen to form the basis of stakeholder analysis in the new, synthesised approach to requirements engineering that is presented in Chapter Six.

3.5 Summary

This chapter has described how important problems with traditional requirements engineering methods are attenuated or eliminated by three selected socio-technical approaches to requirements engineering. However, the socio-technical approaches are themselves associated with significant new problems that might make it difficult for new users to use them: they are described in insufficient detail; they are incomplete in some areas; they are sometimes unclear or even confusing; and they do not define clearly the syntax and semantics of the notations they employ for building models. The
new, synthesised goal-oriented approach to requirements engineering, presented in Chapter Six, documents an attempt to address the new problems. Meanwhile, in the next chapter, the goal-oriented approach to requirements engineering is reviewed. The goal-oriented approach is also relevant to solving some of the key problems with traditional requirements engineering, such as ignoring the high-level objectives of an organisation.
Chapter 4

Deriving requirements from goals
4.1 Introduction

Each of the socio-technical approaches to requirements engineering described in Chapter Three—Checkland’s Information Systems Development [WBPB95], Holtzblatt’s and Beyer’s Contextual Inquiry [HB96, HB98], and Eason’s socio-technical approach [Eas88]—featured, to a greater or lesser extent, the use of current goals and knowledge of current organisational activities to derive requirements for computer-based systems. However, if the goal-oriented and the related process-oriented approaches were fully articulated in these socio-technical approaches, then it is likely that they would become even more effective at tackling some of the problems associated with the traditional approaches, for example at tackling the problem of high-level organisational objectives often being ignored. So the goal-oriented and process-oriented approaches to deriving requirements for computer-based systems need to be fully understood, and this chapter now reviews both. The chapter begins by discussing the rationale for the goal-oriented approach. This is followed by a basic description of this approach. Then existing work on goal-oriented approaches is reviewed, and advantages and problems of the goal-oriented approach are listed. Goals are seen to have a close relationship to processes in that processes are enacted to achieve goals. The chapter ends with a review of process-oriented methods associated with the development of computer-based systems, and, therefore, with the derivation of the requirements for such systems. The reviews highlight the salient features of process-oriented approaches, and surface problems associated with each approach considered. The detailed description of goal-oriented and process-oriented approaches presented in this chapter are used to inform the design of the synthesised approach in the next. And the problems with the goal-oriented and process-oriented approaches encountered here are taken to be problems that need to be addressed by the new, synthesised approach that is presented in Chapter Six.

4.2 Rationale for a goal-oriented approach to requirements engineering

Asking and answering the “why” questions during software development is viewed as important by a number of researchers in the field. For example, McDermid provides a strong justification for this position [McD94]. He first observes that, traditionally, requirements specifications are characterised by the description of a system’s functions. Then, after enumerating a number of problems perceived with such specifications, he says “perhaps most significantly, such specifications may violate the ‘high level objectives’ of the organisation the system is intended to serve” [McD94]1. The violation occurs because “the process which leads to such specifications is effectively ‘starting at the wrong place’, that is with the derived requirements not fundamental ones. By this is meant looking at the desired properties at a presupposed system boundary instead of starting with business needs or objectives of the embedding system” [McD94]2 (thesis author’s italics). In other words, McDermid is saying that much system development starts with too

---

1Page 25
2Page 26
narrow a system boundary; and that this is unhelpful. Instead, it should start by considering the goals of the business or embedding system, in other words the goals of a served system, in Checkland’s terms [WBPB95].

McDermid’s position is shared by other researchers [LK95a] [DBCS94] ³. For example, Loucopoulos and Karakostas state that “it is important to establish at the outset the significance that requirements have in an organisational context” [LK95a] ⁴ (thesis author’s italics). His justification for this assertion is that “understanding the organisational setting is crucial to developing a more complete understanding of requirements for Information Systems. Information Systems and their formal descriptions exist for some reason—they serve strategic, tactical, and operational objectives of the enterprise” [LK95a] ⁵.

So, Loucopoulos asserts that the purpose of Information Systems is to serve objectives of an enterprise, and McDermid warns that if account of them is not taken during systems development, then they are liable to be violated. Thus, for both, the starting point of systems development is the determination of the goals of a served system. But how are such goals determined and represented, and, more importantly, how are they used to produce requirements for computer-based systems? The next section attempts to answer these questions.

### 4.3 The goal-oriented approach to requirements engineering

The most important feature of the goal-oriented approach is that it can be used to derive requirements for computer-based systems from the goals of a served system, for example an organisation or department. In addition, requirements derived in this way are considered to be superior to requirements derived from more orthodox approaches. This is because the computer-based systems implemented from them will, at worst, be unlikely to violate the high-level goals, and, at best, will conform well to them.

In one of its simplest forms, the approach involves decomposing high-level goals to lower-level sub-goals, and then decomposing these in turn, until leaf-goals are reached that express requirements for computer-based systems [LK95a] ⁶. This process is illustrated in the following example adapted from [LK95a] ⁷. Consider the diagram in figure 4.1. It shows how the high-level goal “Increase Profits” has been successively decomposed through three levels to the leaf-goals “Automate Task XYZ” and “Automate Task PQR”. Both these leaf-goals express requirements for computer-based systems. So one high-level goal has produced two possible sets of requirements for computer-based systems that will achieve it. This illustrates an important feature of the goal-oriented approach: it normally produces a number of alternative sets of requirements for computer-based systems.

The preceding paragraph has introduced a number of concepts: goal, sub-goal, leaf-goal, goal decomposition, and alternative sets of requirements.

---

³Pages 93 - 94  
⁴Page 2  
⁵Page 2  
⁶Pages 45 - 47 and 84 - 85  
⁷Pages 45 - 47
These are now discussed in more detail. Goals are usually defined in terms of states to be reached, maintained, or avoided (see, for example, [McD94]). Loucopoulos provides a good definition [LK95a]: “A goal is defined as a defined state of the system. Since a state is described in terms of the values of a number of parameters, a goal can be alternatively defined as a set of desired values for a number of parameters”. Loucopoulos provides the following example of a goal: “To make profits of $1M in the next financial year”. Here, the two goal parameters are “profits” and a time-frame, and the desired values are “$1M” and “within the next financial year” respectively.

Both in the literature and in the world of work, there are a number of common synonyms for the term goal. These include: objective, aim, end, target, and purpose. Often the terms “end” and “objective” are used to stand for goals that are either vaguely expressed or long-term or to be achieved by multiple actions (see, for example, [McD94]). Another term that is often used instead of goal is “constraint”. However, constraints and goals are taken here to refer to different phenomena as will be explained in the sequel.

In the goal-oriented approach to deriving requirements for computer-based systems, high-level goals of a served system are first identified. These are then decomposed successively in order to derive requirements for computer-based systems [DvF93, McD94, LK95a] [AP98]. The notation that is often used to capture the high-level goals and the result of their decomposi-

---

8 Page 28
9 Page 44
10 Page 27
11 Page 158
CHAPTER 4. DERIVING REQUIREMENTS FROM GOALS

Figure 4.2: Goal lattice - a fragment

The basic idea is described and exemplified in Loucopoulos et al. [LK95a]\(^{14}\). A goal may be decomposed in two main ways. Either a goal, G1 say, may be decomposed into a set of conjunctive sub-goals, all of which must be attained in order to attain goal G1. Or it may be decomposed into a set of disjunctive sub-goals, at least one of which must be attained in order to attain goal G1. These ideas are illustrated in figure 4.1. The goal “Increase Profits” is attained if either of the goals “Increase Sales” or “Reduce Production Costs” is attained. And the goal “Reduce production Costs” is attained only if all of the sub-goals “Reduce Cost of Raw Material”, Reduce Cost of Machinery”, “Increase Productivity”, and “Reduce Staff Cost” are attained.

Diagrams like the one in figure 4.1 are referred to as goal hierarchies. However, goal lattice would be a more appropriate name [LK95a]\(^{15}\), as, in general, a sub-goal may support the attainment of more than one high-level goal. This is exemplified in the fragment from a goal lattice shown in figure 4.2. It was taken from a goal-hierarchy that was generated in the case study described in Chapter Seven. The case study describes how a goal-oriented approach is used to generate requirements for computer-based systems that are intended to improve the working of a university faculty help-desk. Recording the details of all problems reported to the help-desk by the users helps to attain the three higher level parent goals that are shown. For example, recording problem details helps to minimise the occurrence of “lost” problems, in other words problems reported by users for which the help-desk can find no record.

So far, applications of the goal-oriented approach have been considered that have involved neither conflicting goals nor multiple views of a served system’s goals. In practice, the situation is often more complicated. Some sub-goals in a goal-hierarchy usually conflict with others. And different stakeholders frequently have different views about the goals of a served sys-

\(^{12}\)Pages 22 - 23 and 87 - 90
\(^{13}\)Pages 100 and 102
\(^{14}\)Page 45
\(^{15}\)Page 45
tem. Dardenne et al. [DvF93] assert that two goals conflict with each other if they cannot be achieved together. They illustrate this kind of conflict with an example taken from a library system. One goal is “Maintain Long Borrowing Period”. This is defined to mean that a borrower may borrow a book for as long as they need it. Another goal is “Maintain Regular Availability”. This is defined to mean that books may be borrowed only for a fixed period. The two goals cannot be achieved together. The first one allows a book to be retained indefinitely. But, should this happen, the second would probably be violated eventually. A slightly weaker form of conflict is proposed by Loucopoulos and Karakostas [LK95a]. They assert that “mutually conflicting goals affect negatively the attainment of each other”. Thus, when two goals conflict, it is possible that both may be achieved in part. This way of defining conflict is incorporated by Bolton et al. in the GMARC goal-oriented approach [BJT+94] in the “is_undermined_by” relationship between goals. In one of their examples of such conflict, the goal “profits made” is undermined by the goal “refunds” allowed, though both can be achieved together to some extent. In the goal-oriented approach to deriving requirements for computer-based systems from goals it is necessary to analyse goal hierarchies for conflicting goals [McD94, LK95a, DvF93].

Conflicting goals need to be identified in order to fully evaluate the effect of alternative solutions, in other words of alternative sets of requirements for computer-based systems. Once conflict has been identified it may be reduced or eliminated by various means. For example, Dardenne et al. propose favouring higher priority goals over lower priority goals [DvF93]. Another way is to encourage stakeholders to negotiate the definitions of conflicting goals until partial, contemporaneous achievement of the goals is enabled.

Conflicting goals complicate the simple picture that was presented earlier. This picture is also complicated in practice by the existence of different views of the main goals and, by implication, the associated goal hierarchies for a served system that are held by the various stakeholders [Kee92, LK95a, DBCS94, CH98, WKRG99, Jac91]. For example, Loucopoulos states that “the process of eliciting goals is far from straightforward since the process is likely to involve multiple participants who will hold multiple perspectives on a single domain” [LK95a]. Different stakeholders often hold different views about the main goals of a served system. Frequently the goals in different views conflict with one another. In order to progress a development, the differences between views need to be resolved. “The private goals of members of an organisation (its owners, employees, and so on) must all be considered and harmonised” [LK95a]. However, while some workers seem sanguine about the feasibility of this task, and provide detailed mechanisms for combining goal hierarchies [Kee92].
CHAPTER 4. DERIVING REQUIREMENTS FROM GOALS

and resolving conflict [DvF93][27], others, for example [Jac91], highlight some of the major difficulties for this endeavour in some common organisational contexts.

4.4 Goal-oriented approaches in the literature

The previous section has described the goal-oriented approach to deriving requirements for computer-based systems. It has presented the main concepts of the approach: goal, goal hierarchy, goal decomposition, and the idea that alternative sets of requirements may satisfy given goals. It has also discussed two inter-related factors that complicate the approach: different stakeholders may hold very different views about what are the main goals of a served system, and conflicting goals may exist both within one view and across different views.

This section reviews research that has been reported in the literature on goal-oriented approaches to deriving requirements for computer-based systems. Research has been categorised according to where its authors have taken the boundary of the served system to lie. Different researchers have defined their served system boundary on either a structural basis or a processual basis. The structural categories are:

- Served system scope defined by the enterprise boundary
- Served system scope defined by the application boundary

The processual category is:

- The focus is on processes

Each subsection below discusses work associated with a particular category and presents representative pieces of research in greater depth.

Served system scope defined by the enterprise boundary

A number of projects have taken the boundary of an enterprise to be the limit of the served system, including the following: GMARC [BJT+94], OR-DIT [DBC94], F3 [BNG94, BRLD94], and [Eas88][28]. Loucopoulos and Karakostas overview a number of approaches to enterprise goal modelling in [LK95a][29], where they also provide a characterisation of enterprise goal modelling and a tutorial example. A representative and relatively complete description of the goal-oriented approach to deriving requirements in the context of enterprises is also provided by Loucopoulos and Kavakli [LK95b]. Their fundamental belief is that “the purpose of building a software system is to be found outside the system itself in the enterprise” [LK95b][30], and that, therefore, knowledge of the enterprise should be captured in a formal model and used to derive requirements for computer-based systems. Their interpretation of an enterprise model comprises knowledge about goals, roles, actors, processes, and resources in an enterprise. This knowledge is held in

---

[27] Pages 33 - 35
[28] Pages 83, 85, and 87 - 90
[29] Pages 82 - 87
[30] Page 47

55
CHAPTER 4. DERIVING REQUIREMENTS FROM GOALS

three different views: the technical, the social, and the teleological. The “technical perspective describes a number of different aspects of the business: business processes, flow of information and data across business processes, where resources or organisations are located, where activities take place, etc.” \(^{31}\) The “social viewpoint reasons about policies, structures and work roles; validates and maintains their established order.”\(^{32}\) The teleological viewpoint is deemed the most important: “Goals are recognised as the primary factors that govern and explain the current and potential enterprise configuration” \(^{33}\). The teleological viewpoint is ideally developed in the manner described in the description of the goal oriented-approach given in section 4.3 (see page 51): goals of the stakeholders are determined, along with the constraints of the enterprise; goals are decomposed into more specific sub-goals; this leads to alternative approaches to their satisfaction, each of which must be evaluated; in addition, conflicting goals should be identified and resolved with the stakeholders before the alternatives are evaluated. The paper reviews an application of their approach to an Air Traffic Control (ATC) enterprise. Although an explicit process model for the approach is not given, the one presented below seems to be implicit in the paper:

1. Investigate the enterprise

2. Develop models of the technical and social viewpoints

3. Develop a model of the teleological viewpoint
   (a) Identify goals, constraints, and problems
   (b) Construct goal hierarchies
   (c) Construct a goal graph
   (d) Identify conflicting goals
   (e) Identify goals to automate
   (f) Define automation strategies
   (g) Evaluate automation strategies
   (h) Select best automation strategy
   (i) Define Information System goals
   (j) Define functional requirements to satisfy IS goals
   (k) Define non-functional requirements to satisfy IS goals

Stage-three is illustrated in detail below using fragments of Loucopoulos’s and Kavakli’s example. In step 3 (a) a requirements engineer should identify goals, constraints, and problems. Some of the high-level goals, constraints, and problems of the ATC enterprise are given below:

- High-level goals
  - Safety (management, customers)
  - Efficiency (management, customers)
– Avoiding boredom (controllers)
– Reducing demands on controllers (controllers)
– Minimize risk of aircraft accidents (management, customers)

• Constraints

– Vertical separation between aircraft cannot be less than 2000 feet
– Horizontal separation between aircraft cannot be less than 5 miles

• Problems

– Reduce delays to aircraft
– Manage problems and emergencies

In 3 (b) the requirements engineer constructs goal hierarchies. Each goal, constraint, and problem is treated as a high-level goal, and decomposed into a goal hierarchy. Three of the four resulting higher-level goals are shown in figure 4.3, which also shows part of the goal decomposition of one of them. In 3 (c) the requirements engineer constructs a goal graph by merging together the goal hierarchies. A part of the result is shown in figure 4.4.

In 3 (d) the requirements engineer analyses the goal graph for conflicting goals. Figure 4.4 shows that two conflicting goals have been identified: “automate sequencing and planning” conflicts with “manage problems and emergencies”. In 3 (e) the requirements engineer scrutinizes the leaf goals of a goal graph in order to identify areas which might benefit from partial or complete automation through computer-based systems. Here, two have been identified: “automate sequence planning” and “automate aircraft sequencing” (see figure 4.4). In 3 (f) the requirements engineer describes the automation strategies that are identified in the previous step. Here, “Automate sequencing and planning” will be completely automated. The associated computer-based system “will automatically detect conflicts and divergences and order the aircraft in time and space in the most efficient way” [LK95b] 34. “Automate aircraft sequencing” will be partially automated: the associated computer-based system “manages only aircraft sequencing” while the controller selects among alternate sequences, updates flight plans and so on.

In 3 (g) the requirements engineer predicts the effects of each automation strategy and categorise them on a scale ranging from beneficial to harmful. Here, the completely automatic “Automate sequencing and planning” is likely to minimize the risk of human error. However, it cannot handle abnormal situations. But they can be handled by the partially automatic “Automate aircraft sequencing”. In 3 (h) the requirements engineer chooses as the best automation strategy the one that evaluates as most beneficial in the previous step. Here, “Automate aircraft sequencing” is chosen. In 3 (i) the requirements engineer determines what goals for information systems are motivated by the leaf goals of the chosen automation strategy. Here, the goal “Visualise air traffic scenarios in real-time” motivates the introduction of the following goal for an information system: “The system should display scenarios” (see figure 4.5). In 3 (j) the requirements engineer determines what functional requirements for computer-based information systems are motivated by the information systems goals. Here, functional requirements

34Page 66
Minimise risk of aircraft accident
Automate control process
Decrease risk of human error
Automate sequencing and planning
Automate aircraft sequencing
Continuously monitor
Visualise air traffic scenarios in real-time
Reduce delays to aircraft
Manage problems and emergencies

Figure 4.3: Air Traffic Control goal hierarchies (after [LK95b])
CHAPTER 4. DERIVING REQUIREMENTS FROM GOALS

Minimise risk of aircraft accident
Automate control process
Decrease risk of human error
Automate sequencing and planning
Automate aircraft sequencing
Continuously monitor
Visualise air traffic scenarios in real-time

Figure 4.4: Air Traffic Control goal graph (after [LK95b])
include the following: “Display tracks”, “Display maps and airways”, and so on (see figure 4.5).

In step 3 (k) the requirements engineer determines what non-functional requirements for computer-based information systems are motivated by the information systems goals. Here, the goal “Visualise air traffic scenarios in real-time” motivates the introduction of the following non-functional requirement for computer-based information systems: “The system should perform in real-time”. This abstract requirement, in turn, motivates the introduction of other non-functional requirements, which, in turn, motivate the introduction of others, until the following detailed requirement is motivated: “Display aircraft in < 3/16 sec of sweep period”.

A critical evaluation of Loucopoulos’s and Kavakli’s approach yields two potential problems. First, Loucopoulos and Kavakli seem to assume that there is a close relationship between the goals of an enterprise and the personal goals of its members: “The members of the enterprise work toward the joint enterprise objectives, perhaps different but related to their personal objectives”. However, a number of writers, for example Checkland [CH98] and Jackson [Jac91] challenge this view: they maintain that in many organisations individual members and groups of members have goals that differ both from the official enterprise goals and from each other. And second, although Loucopoulos’s and Kavakli’s approach is very comprehensive, it may be difficult to apply because it is not described in great detail.

Served system scope defined by the application boundary

The boundary of a served system may be taken to encompass a whole enterprise. On the other hand it may be taken to encompass just some part of an enterprise. Parts may be determined on the basis of structural division. In this case, example parts might include, for example, division or function or project or team. Or they may be defined in terms of some service or function required throughout an enterprise. Internal e-mail and meeting scheduling are examples of such parts of an enterprise. A number of goal-oriented approaches to deriving requirements for computer-based systems have been developed that are capable of working with the served system defined as an application service or function with an enterprise-wide scope [Fea87, FH92, DvF93, BJT+94, Chu93]. These approaches are illustrated in [Gre93], where they are applied to the classic “meeting scheduler” problem (see Appendix A). One of these approaches—KAOS—is described again here, since it provides an illustration of the most formal application of the goal-oriented approach to deriving requirements for computer-based systems.

The aim of the KAOS approach [DvF93] is to derive a description of a system’s behaviour and an initial analysis of its structure through acquiring and formalising functional and non-functional requirements (or goals) for a composite system. Figure 4.7 shows the inputs to KAOS.

The meta model is actually an integral part of KAOS, but is included here as an input as it may be changed occasionally. The meta model is like an ERA schema; here it characterises composite systems. As such it comprises meta concepts, for example goal, constraint, agent, action, and so on; meta relationships, for example agent-performs-action, action-ensures-constraint, constraint-operationalises-goal, and so on; meta attributes characterising
CHAPTER 4. DERIVING REQUIREMENTS FROM GOALS

Figure 4.5: Air Traffic Control functional requirements (after [LK95b])

Figure 4.6: Air Traffic Control non-functional requirements (after [LK95b])
both, for example pre-condition is a meta attribute of the action meta concept and strengthened pre-condition is a meta attribute of the ensures meta relationship; and meta constraints, for example constraints defining the cardinality of a meta relationship. Figure 4.8 shows a part of a KAOS meta model. The order and manner in which meta objects are acquired is determined by the meta model traversal strategy (or acquisition strategy). The meta model could be traversed backwards, either from identified agents, or from client supplied scenarios; however, for the remainder of this section, only backwards traversal from a client’s requirements (or goals) is considered. Knowledge about traversal strategies is included in meta knowledge used by a computer based acquisition assistant to guide an actual traversal. The assistant also uses domain knowledge of varying degrees of abstraction in order to perform analogical reasoning.

The initial basic source of information for this approach, using a goal-driven strategy, is a client’s expression of functional and non-functional requirements for a composite system. These are input to the seven-stage KAOS process, which is shown in figure 4.9. During the KAOS process, instances of each of the meta objects discussed above are acquired, as a requirements model is gradually created. During stage-one, the highest level goals of a system are identified from the requirements. Goals are deemed to be system objectives which cannot be met by the actions of just one agent. Such goals are reduced to an equivalent set of objectives each of which can be met by the actions of one agent. These objectives are called constraints and are formally defined during stage-three. The reduction is achieved by decomposing goals into one or more sets of sub-goals, where each member of a set contributes to the satisfaction of a parent goal, and satisfying all the sub-goals in a set completely satisfies the parent goal [Nil71]. If it is possible, goals should be expressed formally using a first-order temporal logic (currently a formalism inspired by ERAE [Dub91] is used by the KAOS group). As the goal decomposition proceeds, objects (agents, entities, relationships, and events) referred to in formal and informal goal descriptions are identified and abstracted for use in stage-two.

During stage-two, objects associated with goals are reviewed, and agents (objects which control state transitions) are identified along with their actions and any pre-conditions, post-conditions, and trigger-conditions for their actions. This knowledge of agents and actions is then used in stage-one to identify when a goal may be reduced to a constraint, in other words to identify leaf-goals. Thus, stages one and two may be viewed as co-routining stages. In stage-three, each leaf-goal of a goal structure is converted into a constraint whose formal definition is expressed in terms of objects and
Figure 4.8: A portion of a KAOS meta model
CHAPTER 4. DERIVING REQUIREMENTS FROM GOALS

Requirements Engineer

Create goal structure
Identify agents
Identify constraints
Identify new objects, actions, and attributes
Identify ensure links for constraints
Assign responsibility for constraints to candidate agents
Assign responsibility for each constraint to selected agent

Client’s requirements available
Requirements model available

Figure 4.9: The KAOS process
CHAPTER 4. DERIVING REQUIREMENTS FROM GOALS

In stage-four, new actions and objects described in the formal definitions of constraints during stage-three are identified and acquired. In addition, new characteristics of already acquired concepts, such as new meta attributes or new pieces of invariants, pre-conditions, post-conditions, and trigger-conditions are also identified. In general, actions, pre-conditions, post-conditions, and trigger-conditions identified in stage-two are not necessarily sufficient to ensure that each constraint will be satisfied. So, in stage-five, each constraint is examined and consequently, to ensure that each constraint will be met, some actions and objects may be modified, and new actions and objects may be introduced. Such modifications might involve strengthening an action’s pre-condition, for example, or strengthening an object’s invariant. This new information is held in a requirements model as meta attribute values of the ensures meta relationship. In addition, new actions are acquired which allow soft constraints (see below) which may have been violated to be restored.

In a complete requirements model each constraint is the responsibility of a single agent. During stage-six, each constraint is reviewed in turn, and agents which might be made responsible for it are linked to it by instances of the responsibility meta relationship. Finally, in stage-seven, the actions required to satisfy a constraint are assigned to the “best” agent from among the candidate agents for satisfying that constraint.

The rationale for the meta model and the acquisition process is as follows: the system goals correspond to a client’s requirements; these are satisfied if all the leaf goals of the generated goal structure can be satisfied; these, in turn, can be satisfied if the constraints operationalising them can be met. The formal definition of each constraint defines a set of sequences of states in terms of properties of objects in the states, that is to say in terms of patterns of states. In order to move from one state in a sequence to the next, actions must be performed on one or more of the objects in the input state. In general, a set of actions (with appropriate pre-conditions, post-conditions and trigger conditions) acting on objects (conforming to appropriate invariants) will be needed to meet a constraint. A number of agents may be able to perform actions which will enable them to satisfy the constraint. One of these is selected as the agent which is responsible for meeting the constraint; this one has the appropriate actions assigned to it. The approach is used to create a requirements model for a composite system. This model comprises, among other things, a set of agents, where each agent has a set of actions and a set of constraints for which it has been assigned responsibility. Different constraints may conflict with one another.

The KAOS approach is illustrated here by applying it to the Meeting Scheduler problem (see Appendix A). The initial basic source of information for this approach, using a goal-driven strategy, is a client’s expression of functional and non-functional requirements for a composite system. For example, for the meeting scheduler system, such goals include the following: “to determine, for each meeting request, a meeting date and location so that most of the intended participants will effectively participate”, and “privacy rules should be enforced; an ordinary participant should not be aware of constraints stated by other participants”.

A client’s requirements are input to the seven-stage KAOS process (see
Figure 4.10 shows the results of partially decomposing one of the meeting scheduler system’s functional goals during stage-one. From complete goal descriptions (not given) for the goals referred to in figure 4.10, it is possible to identify from the Schedule Meeting goal the following objects: meeting request, date, location, meeting scheduler, attendee (in other words meeting participant); and actions: participate. During stage-two, these objects and actions are reviewed, and Attendee and Meeting Scheduler are identified as agents, and Participate (at meetings) is identified as an action of the Attendee agent. Similarly, an Authorised User agent, and Request Meeting and Accept Meeting actions are also identified during stage-two.

Continuing the example, in stage-three, the Attendees’ Preferences Known system goal is operationalised into two constraints: a hard constraint (hard constraints must not be violated), Request For Attendees’ Preferences Made, and a soft constraint, Request For Attendees’ Preferences Satisfied. This is shown in figure 4.11. The first constraint may be satisfied by a Request Attendees’ Preferences action available to the Meeting Scheduler agent; the second may be satisfied by the Submit Preferences action available to Attendee agents. It is assumed that both actions were acquired from the client during stage two.

In stage-four, it is determined that the two constraints in the example contain no new objects or actions, nor do they contain anything which completes the description of objects and actions already identified. In stage-five it is recognised that Request For Attendees’ Preferences Satisfied is a soft constraint which signifies that all the requests for attendee preferences will be satisfied within a predefined time period. If the constraint is violated, actions are required to help to restore it. One such action might be Send Preference Request Reminder; in the meeting scheduler requirements model, this action would be linked to the constraint by an instance of the restoration meta relationship.

Stage-six and seven are concerned with allocating agents. In the example, the Attendee agent is linked by a responsibility meta relationship to the Request For Attendees’ Preferences Satisfied constraint, and the System Scheduler agent is similarly linked to the Request For Attendees’ Preferences Made constraint. Had other agents been identified earlier which might also have been made responsible for either of these constraints, then they would also have been similarly linked. Finally, in stage-seven the Request For Attendees Preferences Made is assigned to the Meeting Scheduler agent by an instance of the performs meta relationship link. Figure 4.12 shows a fragment of the requirements model acquired for the meeting scheduler system by the approach just described.

Critical evaluation of the KAOS approach identifies two potential problems: the lack of tool support for KAOS combined with its use of formal notations (temporal logic) may make non-experts reluctant to use it in real-world organisational contexts.

4.5 Advantages and problems of the goal-oriented approach

The main advantages of the goal-oriented approach to requirements engineering have been summarised by Axel van Lamsweerde [vL01]. First, the
Figure 4.10: Partial decomposition of a meeting scheduler system goal
goal-oriented approach is characterised by a systematic requirements elaboration process: “Goals drive the identification of requirements to support them.” Second, the approach facilitates the detection and resolution of intrinsic conflict: “Goals have been recognised to provide the roots for detecting conflict among requirements and for resolving them eventually.” Third it facilitates the exploration of alternative “solutions”: “Alternative goals provide the right level of abstraction at which decision makers can be involved for validating choices being made or suggesting other alternatives overlooked so far.” Looking for alternative solutions means that more solutions, and, in particular, more radical solutions are often found. Fourth, the approach is associated with achieving requirements completeness: “Goals provide a precise criterion for sufficient completeness of a requirements specification.” Fifth, goals may provide a rationale for requirements: links from requirements to goals may be used to explain and justify requirements to stakeholders. Sixth, the goal-oriented approach helps the requirements engineer and the stakeholders to avoid irrelevant requirements: “Goals provide a precise criterion for requirements pertinence.” And seventh, goals may help to structure requirements documents: “Goal refinement provides a natural mechanism for structuring complex requirements documents for increased readability.”

In addition to the advantages identified by van Lamsweerde, there are other advantages. First, the goal-oriented approach may facilitate the understanding of a system. The goal-oriented approach generates a set of goals for a system, and reading goal descriptions may help to provide a high-level understanding of the purpose and nature of the system. Second, the approach may facilitate evaluation of a system, since goals constitute criteria against which an implemented system may be judged. Third, the approach encourages implementation detail independence by making goals (relatively abstract), rather than activities (relatively concrete), the centre of focus.
Figure 4.12: Fragment of a requirements model for a meeting scheduler system
Fourth, it facilitates the provision of re-usable structures: established goal structures may be re-used in similar domains; see, for example, GMARC in BJT+94. Fifth, the approach provides, in goals, a starting point for traceability links between all the artefacts created during the development of a piece of software GF96 and Bir99. Sixth, the approach may facilitate the evaluation of designs: a design may be judged according to how well it satisfies a set of goals Chu91. And seventh, the approach may be used to facilitate maintenance: in an approach developed by Karakostas Kar90 requirements that need changing “point to” the system components that may be impacted by the change.

However, although the goal-oriented approach has a number of advantages, there are also a number of potential problems associated with it. For example, one problem is associated with decomposing goals: how can one be sure that a goal is precisely satisfied by its sub-goals? Another problem related to the difficulty that sometimes arises over distinguishing goals from solutions: “Distinguishing between primitive goals (leaf-goals) and the means to achieve them is not always clear” LK95b. While a third problem concerns the dynamic nature of many organisations. In such organisations the goals may change so quickly that the goal-oriented approach is rendered less useful.

### 4.6 Goals and processes

This section reviews work that exemplifies the process-oriented approach to requirements engineering, an approach that focuses upon processes as well as upon goals. A process is taken to comprise a collection of activities undertaken by one or more agents (people or computer-based systems) some of which involve interaction between the agents. Processes are normally intended to achieve goals, in other words the activities of a process are normally intended to produce a state corresponding to a goal to be achieved. Process models are used to describe processes. They may represent both the activities of a single agent, and activities involving two or more agents.

Once a process has been characterised and expressed as a process model, a number of uses may be made of it. First, it may be analysed to see, for example, how well it meets its intended goals; second, it may be modified in order to try to produce process performance improvements. Provision of IT support for a process constitutes one kind of modification. Appropriate IT support, and thus the requirements for it, may be determined in at least two distinct ways: either the existing process may be analysed and and areas which might benefit from IT support may be identified (see figure 4.13); or, alternatively, the goals of a process might be reviewed in parallel with an appraisal of potentially relevant IT. Following this, a new process could be designed that would exploit the latest IT in order to achieve the goals (see figure 4.14). One or other variant of this method is found in recent work on Business Process Engineering (BPR) Dav93, Ham90, HC93, Oul95. It is also found in more recent work that focuses upon providing computer-based support for business processes WKRG99. Three process-oriented approaches are reviewed here, with a view to surfacing ideas that might be incorporated in the synthesised approach to
goal-oriented requirements engineering that is described in the next chapter. The three approaches are: Davenport’s [Dav93], Ould’s [Oul95], and the approach of Warboys and his colleagues [WKRG99]. The reviews also identify problems with each approach that the new, synthesised approach is intended to eliminate or attenuate.

### Davenport’s approach

Among the BPR initiatives, Davenport’s [Dav93] method conforms closely to the first variant of the goal-oriented method outlined above. Davenport states: “The objectives of business process innovation fall into three groups, which tend to lead into one another: strategic objectives lead to process-related objectives, which lead to information (computer-based) systems-
CHAPTER 4. DERIVING REQUIREMENTS FROM GOALS

related objectives” [Dav93] 36. Davenport has designed a method for deriving requirements for IT process support from captured strategic objectives. This method is first presented in outline, and then elaborated.

• Identify processes to innovate
• Identify IT enablers for new process design
• Define business strategy and process vision
• Understand the existing process
• Design and prototype the new process

For the first stage, Davenport suggests that the processes to innovate could be chosen informally, based upon, for example, their perceived centrality in relation to the firm’s business strategy or upon their health, in other words upon whether or not they were exhibiting serious problems. He also sketches a more formal procedure for selecting processes to innovate: first, identify and prioritise the goals of an organisation; then identify its main processes; and, finally, choose process(es) to innovate based upon the number and importance of the goals that they support.

For stage-two, Davenport recommends that, before undertaking process redesign, a process redesigner should be aware of the range of IT tools that have potential for supporting the process: “A process designer pursuing innovation should consider all the tools that can be used to shape or enable the process, and IT and the information it provides are among the most powerful” [Dav93]. Knowledge of the potential of such IT tools may then actually influence design decisions. This method is arguably better than the method in which a process is redesigned in the absence of such knowledge, followed by a search for IT tools to support the resulting process. Better because, in the former case, potentially applicable, powerful tools may be more easily incorporated into the process, and, in the latter, they are much less likely to be. Davenport suggests three mechanisms that are intended to facilitate the selection of potentially useful IT tools:

• Maintain an index of IT-supportable areas
• Map IT tools to generic business areas
• Carry out benchmarking, in other words examine how the same or similar processes have been implemented in other organisations or contexts

Stage three of the approach is concerned with defining a business strategy and process vision. Davenport identifies a series of “drivers” for his method—strategy, process vision, process goals, actions—where each driver is used to derive the next one in a series. So, the key driver is the first one: a well-defined strategy. It has already seen that this may be used to select a process to innovate. Here it can be seen that it may be used in “the development of process visions for processes to be innovated”. For Davenport, a strategy and a process vision are closely related phenomena: but “we
view strategy as long-term directional statements on key aspects of a firm or business unit, and vision as a detailed description of how, and how well, a specific process should work in future”. Strategy includes process-oriented, product-oriented, and financial goals that are measurable and to be achieved in a time-frame of five to ten years. On the other hand, a process vision which is derived from a strategy comprises the main goal of the process and the other process objectives, preferably quantified, and is to be achieved in a shorter time-span. A typical process objective might be: “reduce new drug-development cycle time by 50% in three years”.

The existing process should be characterised in stage-four. This involves both modelling its structure and flows, and (ideally) measuring its performance. For Davenport, the main reason for the former is to understand the cause of a process’s problems. The main reason for the latter is to make performance comparisons between the old and new process.

When stage-five is reached, a process re-designer should have knowledge about the current process, a vision for the new process, and IT tools offering the potential to support the new process. Guided by the process vision, the process redesigner redesigns a new process. He or she includes in the design appropriate IT tool support and thus produces, if only in embryonic form, descriptions of the requirements of such tools. Davenport recommends that the new process should be prototyped and the prototype tested by the users. Such prototypes are intended to include prototype versions of the supporting IT tools.

Critical evaluation of Davenport’s approach yields potential problems that might make it difficult to apply. First, it is worth noting that each of the three mechanisms in stage-one is based, at best, upon the current manifestation of a process (including its goals). However, a new process is not designed until later, and its constituent activities might be different from the activities of the current process. But IT tools chosen for their relevance to the current process manifestation might not prove to be so relevant to a redesigned process. Second, it is worth noting that since a strategy is used to select processes to innovate, then strategy definition should be performed prior to the first step, Identify Processes to Innovate. Third, the approach’s steps are not expressed in sufficient detail to allow unambiguous interpretation. And finally, Davenport has not provided a process modelling language. The third problem is tackled in the approach synthesised here (see next chapter). The latter problem has been addressed by a number by researchers who have created languages or notations for modelling processes: activity diagrams in UML [Fow97], and Role Activity Diagrams (RADS) [Oul95].

Ould’s approach

Two criticisms of Davenport’s method—insufficiently detailed process steps, and lack of a process modelling language—are addressed successfully by Ould [Oul95]. Ould’s method is worked out in more detail than Davenport’s, and, in addition, Ould provides a language for modelling processes: Role Activity Diagrams (RADs) (See Appendix C for details of the RAD notation). Ould [Oul95] presents an eight stage model. Here, for simplic-
CHAPTER 4. DERIVING REQUIREMENTS FROM GOALS

ity, a condensed version of his method is presented. It has four main stages, as follows:

- Decide on the objectives of modelling
- Investigate and model the current process of interest
- Analyse the current process
- Respond to the analysis

This method may be specialised in a number of ways depending upon the objectives of the modelling. However, here, the interest is in both improving the current process, and in providing IT support for either an existing or an improved process. Hence, in this case, the method given above specialises to the following:

- Investigate and model the current process of interest
- Analyse the current process in order to improve it
- Support the improved process with IT

The details of these stages are given in [Oul95]38. In the first stage various sources of information—people, documents, terms of reference, meetings—are accessed. For example, people involved in the process are questioned about it individually and in groups. The following kinds of information is elicited:

- Process goals
- Roles
- Interactions
- Triggers
- Process problems
- Suggested process improvements

In stage-two, the current process of interest is analysed with a view to improving it. The most important mechanism for achieving improvements is based upon a movement from the current concrete process to an abstract process, then another movement to a new concrete process. First, one abstracts away from the “concrete” current process and produces an “abstract” process. As Ould notes: “in an abstract model we look at intent or purpose”. In other words, the abstract model is providing a view of the goal(s) of the process. From the abstract process model, a new concrete process is designed. The differences between the original and new concrete processes may include changes to individual actions, changes to interactions, changes to role structure, or changes to the assignment of roles to parts of the organisation.

In stage-three, the improved process is examined to see which of its areas can be supported by IT, and which of these areas is amenable to which sort of technology. Ould identifies three basic kinds of IT support:

38Chapters Nine and Ten
• Traditional information systems for supporting information needs of individuals

• Workgroup computing for supporting soft complex interactions between people

• Workflow management for supporting the overall process and flow of activity

Ould also provides heuristics for generating requirements for supporting computer-based systems that support an improved process:

• Review black-box activities in RADs to see whether they may be:
  – fully automated
  – performed by humans with computer-based system support

• Review interactions in RADs to see whether they may be supported by:
  – workflow management systems
  – workgroup computing systems

Critical evaluation of Ould’s process-oriented approach yields two potential problems. First, the new process is derived from the abstract form (goals) of the current process. It seems that both users’ problems with the current process, and their new goals for the new process, although perhaps elicited during the “investigate and model the current process” stage, are not explicitly used to derive the new process. Second, whereas in Davenport’s method, IT tools that might support a new process are identified either before or while the new process is being designed, in Ould’s method, IT tool support is determined after the new process has been designed. But, as has been already observed, because the design of the process is not influenced explicitly by potentially appropriate IT tools, it seems less likely that the most appropriate tools, once identified, will be easy to integrate into the process. In the approach presented in the next chapter, both of these criticisms are addressed. It explicitly elicits stakeholders’ views of the current process problems and new goals, and uses both in developing a new process. Second, like Davenport, it suggests that the identification of appropriate IT tools and the design of the new process should proceed in parallel.

Organisational Process Modelling (OPM)

Kawalek et al. [WKRG99] have developed the Organisational Process Modelling (OPM) approach to deriving software. Organisational Process Modelling focuses upon goals and processes and is used to derive requirements for computer-based systems intended to support organisation processes. Such computer-based systems, termed active models, are designed to coordinate the actions of people and software tools. They are also intended to coordinate interaction between people, between tools, and between people and tools. Thus such active models are intended to guide and control process enactment within organisations.
The work of Kawalek et al. extends that of Ould through both the introduction of new models in addition to the Role Activity Diagram models, and the introduction of new notation to the existing Role Activity Diagram notation. Kawalek et al. also focus mainly upon the production of active models, unlike Ould, who focuses equally upon workflow systems, groupwork systems and traditional IT support (database and personal productivity tools). Indeed, Kawalek et al. have designed a language, the Process Modelling Language (PML) for specifying active models.

Organisational Process Modelling conceptualises organisations in terms of people who interact both with each other and with software tools in order to achieve organisational goals. Through considering the interactions between people, the goals that they are seeking to carry out, and the activities that they carry out, a user of Organisational Process Modelling identifies software systems intended to support the behaviour, and thus to achieve the goals [WKRG99]39. Organisational Process Modelling “is concerned with the description of operational goals of users and their mapping to technical capabilities. Thus, it seeks to describe why people collaborate, and how their goals are fulfilled”[WKRG99]40.

Figure 4.15 summarises Organisational Process Modelling in a Role Activity Diagram. Using the notation’s magnifying glass symbol for identifying goal states, the diagram shows that the output from the Organisational Process Modelling process is an active model. This comprises one or more computer-based systems (and therefore their corresponding requirements) that are intended to support organisational processes. The support constitutes managing the actions and interactions of people and tools participating in the processes. The active model is produced during the last stage of Organisational Process Modelling, that is to say during the stage called “Design IT support”. The “Design IT support” activity works upon both a description of the organisational context and problems encountered in that context in order to derive an active model. The organisational model is expressed in terms of three different models: a system (role) model, a goal model, and one or more method models. The first represents people working in context; the second their organisational goals; and the third, one or more ways through which the people might achieve their goals. The latter model is expressed in an augmented Role Activity Diagram.

The first stage, Investigate Context, focuses first upon the investigation as an intervention in the organisation, and second upon the process of interest. The intervention will probably take the form of a project. Issues of concern might include, for example, timescales, budget, resources, aims, scope, deliverables, etc. The focus upon the process of interest involves first of all investigating the process—its objectives, boundary, inputs and outputs, participants, performance problems, and so on. Second, it involves a thorough capture and analysis of the objectives of the process [WKRG99]41. Methods used to capture and analyse the process objectives are listed below. The main outputs of stage-one are details of the process of interest: its

---

39 Pages 4, 11, 15, 68, and 72
40 Page 68
41 Page 92
Figure 4.15: Organisational Process Modelling: a Role Activity Diagram model

Stage-two of Organisational Process Modelling is concerned to identify interacting agents, to describe their goals, and to describe how the goals may be achieved. Three different models—system model, goal model, and method model—are used to model these different aspects of the process of interest. The first model, the system model, depicts interaction between agents. For example, in an organisational context where IT users report IT problems to a Helpdesk, the role model shown in figure 4.16, might be obtained. It shows that in general a number of User agents, or people in other words, interact with a single Helpdesk agent, a person or a group of people.

The second model, the goal model, shows what goals are obtained by each agent in a system model. Continuing the example, figure 4.17 shows the goals that are obtained by the User and the Helpdesk agents respectively. It shows that the User agent initiates interaction with the Helpdesk agent
in order to obtain a solution to an IT problem, in other words to achieve the goal “solution obtained”. The Helpdesk agent’s part in the interaction is complete when a solution has been communicated to a user, that is to say when the goal “solution given” has been obtained. This goal model also expresses the non-functional goal that the “solution given” goal should be obtained within two weeks.

The third model, the method model, shows one way in which the goals of interacting agents might be achieved. The modelling formalism is a superset of the notation used in Role Activity Diagrams [Oul95]. Completing the example, figure 4.18 shows how the “solution obtained” and “solution given” might be achieved. It shows that for the User to achieve the goal “solution obtained”, they must first communicate the problem to the Helpdesk agent. That agent logs the problem, solves the problem, and communicates the solution back to the User. This activity achieves both the User’s and the Helpdesk’s goals of interaction.

The main outputs from the stage are the system, goal, and method models. These are input to stage-three along with the context problems uncovered during stage-one.

The purpose of stage-three is to create software that will support the coordination of interactions between people, between tools, and between people and tools, as they participate in enacting a process. But which parts of a process should receive such support? In Organisational Process Modelling, process problems that are identified during stage-one (Investigate context) are used to identify parts of the process to investigate for process support by IT. Following reflection upon a process problem and upon the models
(system, goal, and method) associated with that problem, the designers suggest ways of eliminating or attenuating the problem using IT support, and in particular workflow techniques. The solution is represented in a new method model. This model shows just those roles relevant to solving the problem. The IT solution is shown on the model as one or more annotations. The final stage is to use the new method module to design PML code to implement the IT solution featured in the model. Later the PML code may be compiled to create (part of) an active model that will support the process in the intended way. The Role Activity Diagram diagram in figure 4.19 summarises the description just given.

The description above illustrates what Kawalek et al. call refinement. Refinements involve planned change in order to solve problems or to find new ways of meeting existing goals. Kawalek et al. also note that two other categories of change might occur during stage-three, viz reflections and re-inventions. Reflections are the relatively small, unplanned changes that people suggest might be made to processes. They are the kind of ideas that manifest themselves spontaneously when processes are being investigated, talked about, and modelled. Re-inventions, on the other hand, involve either major change that “substantially eradicates existing patterns of behaviour” [WKRG99] or the introduction of a process and its IT support into a “green-field” situation.

Critical evaluation of Organisational Process Modelling yields three potential problems. First, during process capture, it is not clear how the goal hierarchies captured in stage-one are used. In particular it is not clear what use is made of higher-level goals. Second, stage-two seems to model the current organisational processes, and stage-three produces IT support for them (active model). This may lead to the approach providing IT support for just
the status quo, rather than for improved or new organisational processes. The latter seems more desirable. And third, stage-three focuses on the goals of agents doing the work. It seems to largely ignore higher-level business goals of other agents, for example, managers and process owners. Each of these potential problems is addressed in the synthesised goal-oriented approach that is presented in the next chapter.

4.7 Summary

This chapter has reviewed goal-based approaches to requirements engineering and process-based approaches to requirements engineering. Problems with all the approaches considered have been identified. The next chapter presents a generic model for goal-oriented requirements engineering that integrates the socio-technical approaches and goal-oriented approaches.
Chapter 5

Relating the socio-technical and goal-oriented approaches
5.1 Introduction

Chapter Five begins with a discussion that relates the goal-oriented approach to requirements engineering, presented in the previous chapter, to the three socio-technical approaches to requirements engineering that are presented in Chapter Three. The discussion notes, in particular, the presence of key features of the goal-oriented approach in the three socio-technical approaches, albeit in embryonic form. The section concludes by presenting a generic model of goal-oriented requirements engineering that is based upon a consideration of both approaches. (This generic model is fully developed and elaborated in Chapter Six as the new, synthesised goal-oriented approach to requirements engineering.) In the second part of the chapter, two approaches—ISAC [LGN81] and CACO [FTB94]—are presented, which may be viewed as early manifestations of the generic model. Shortcomings of both approaches are described.

5.2 A discussion and generic model

Chapter Three describes in detail three socio-technical approaches to requirements engineering: Checkland’s Information Systems Development method [WBPB95], Holtzblatt’s and Beyer’s Contextual Inquiry [HB93, HB96, HB98], and Eason’s socio-technical approach [Eas88]. Each one is seen to eliminate problems associated with methods based upon the traditional requirements engineering process. In this section, similarities in structure between the three approaches are identified and analysed, and their relationship to the goal-oriented approach is articulated. This leads to the creation of a generic model for goal-oriented requirements engineering. In Chapter Six, this model is developed into the new, synthesised approach to goal-oriented requirements engineering.

The main principles behind each approach are very similar; they vary only in specificity. Checkland’s Information Systems Development is based upon the principle that the served system should be investigated before the serving system. But the concept “served system” is intrinsically flexible or elastic: in different cases, a served system may be taken as having a boundary either close to the serving system, or distant from it. The served system concept does not comprehend a pre-defined limit. In Eason’s approach, the served system may be viewed as being limited first to the boundary of the organisation, then to boundaries around selected parts of the organisation. Contextual Inquiry’s main initial activity (observation of users) further restricts the served system to be those users who are likely to be direct beneficiaries of any new computer-based systems. However, another part of Contextual Inquiry—the development of the context model—suggests that aspect of a wider served system are investigated, although it does not indicate how this should be achieved.

So each approach focuses first upon the served system. But what, in detail, does focusing upon the served system mean? What information about the served system is deemed to be important enough to capture? How is it captured? Why is it deemed important? How is it recorded?

The main foci of the three approaches is upon objectives, or goals, of a served system, and users’ work within a served system. They all focus
upon understanding the users’ work, and two approaches (Checkland’s and Eason’s) also focus on the goals or purposes of an organisation, while even Contextual Inquiry explicitly identifies and uses the motives for users’ work. Let us consider objectives first. Eason recommends that goal hierarchies should be developed using top-down decomposition, first for the whole organisation, then for relevant parts of it. Checkland recommends that the purposes of an organisation (or of part of an organisation), as variously seen by its stakeholders, should be captured and expressed in one or two paragraphs of pithy text. In Contextual Inquiry, there is a notion of motive for each aspect of users’ work: motives need to be captured and expressed in order to complete the design of new work. So it can be appreciated that the important concepts of the goal-oriented approach are already present in the socio-technical approaches to varying degrees.

Now let us consider the users’ current work practices. Contextual Inquiry uses observation and questioning to capture the users’ current work. The results are represented using six different types of work models. In Checkland’s approach, the users’ work is captured through informal observation and interview. It is recorded using Human Activity System models. Human Activity System models are like process models: they depict sequences of actions (verbs) on objects (nouns). Eason’s approach recommends that the current organisation structure, the current work, and the current use being made of technology should all be captured. Understanding the users’ current work is important in each approach, both because such knowledge can help to derive the goal hierarchies mentioned earlier, and also because it expresses how such goals are currently being met. It is important to know this because any future situation will have to be reached from the present state, so, to some extent, the present state usually determines possible future states.

So, the three approaches take an organisation’s objectives, or goals, and its users’ current work to be important features, to be captured and recorded. But why is this domain knowledge needed by each approach, and how is it used? Each approach uses objectives (purposes, goals, motives) and users’ current work, directly, to derive new work practices, and, indirectly, to identify, and derive requirements for, computer-based systems that support the new work practices. This pattern can be seen most clearly in Checkland’s Information Systems Development approach. First, various views of the purposes of a domain are described. From these purposes, theoretical models of new work practices—Human Activity Systems models—are derived such that the new work practices achieve the purposes. These theoretical models of new work practices are compared by the analysts and stakeholders to the current work practices. Following debate about the results of the comparison, a model of new work practices that accommodates the different viewpoints of the various stakeholders is agreed upon. This model of new work practices is then analysed both to identify where support from computer-based systems would be feasible and appropriate, and to identify the requirements for such systems.

So, focusing first upon the served system entails understanding both the objectives and current work of a domain. Using this knowledge, new work practices and the requirements for computer-based systems to support them are derived. The next step in each approach is to evaluate the effectiveness
5.3 Early attempts at integrating the socio-technical and goal-oriented approaches

A number of approaches have been designed that are similar to the approach described in the generic model. These approaches include ones that have
been considered already: Eason’s socio-technical approach [Eas88], Checkland’s Information Systems Development method [WBPB95], and Holtzblatt’s and Beyer’s Contextual Inquiry [HB93, HB96, HB98]. They also include those in the list below. Each approach in the list is now presented, and compared to the generic model.

- Information Systems Work and Analysis of Changes (ISAC) [LGN81]
- Concept Analysis and the Concept of Operations (CACO) document [FTB94]

**Information Systems Work and Analysis of Changes (ISAC):**

ISAC is a method for developing information systems to support organisational activities. It was developed during the 1970s by Lundeberg et al. [LGN81]. ISAC has four main stages:

- Change Analysis
- Activity Study
- Information Analysis
- Implementation

The first two stages of ISAC are well described and illustrated by Wieringa [Wie96]. The first stage, Change Analysis, has ten steps:

*Make a list of problems:* elicit problems; associate problems with stakeholder groups; reduce the list by eliminating trivial and impossible-to-solve problems

*List problem owners:* draw a problem matrix showing which stakeholder groups are associated with which problems

*Analyse the problems:* draw a cause-effect graph to try to identify underlying problems; try to quantify the problems

*Make an activity model of the current business:* show on the activity model activities and information flows

*Analyse goals:* elicit stakeholders’ goals for future work activities; identify goal conflict and try to resolve it using negotiation

*Define change needs:* form into clusters problems related to a particular goal; each cluster is a change need

*Generate change alternatives:* Stakeholders brainstorm alternative ways of satisfying change objectives

*Make activity model of desired situation:* aggregate change alternatives into packages; model each package with an activity model

*Evaluate alternatives:* evaluate each alternative by assessing how well it solves current problems and what new problems it creates and for whom they are problems
Choose an alternative: the developer suggest an alternative, but the client has the final choice

The chosen alternative may or may not involve information systems. If it does not, ISAC is exited at this point. If it does, the remaining three stages of ISAC are enacted as previously mentioned. During these stages the information systems which will be automated are identified and their functionality is specified. For example, the second stage, Activity Study, uses the results of the first stage to identify required (automated) information systems, to specify non-functional requirements for them, and to begin to specify their interfaces and functionality. This latter activity is completed in stage three.

Comparing ISAC with the generic model, the first stage, Change Analysis, is similar in that it focuses upon the way an organisation currently works. As an additional similarity, goals for future working are agreed during this stage. However, unlike the generic model, the elicitation and modelling of current objectives is not required by the ISAC approach.

Concept Analysis and the Concept of Operations document:

Concept Analysis to produce a Concept of Operations document is a method developed by Fairley et al. [FTB94]. The method is intended to facilitate the production of more effective and more efficient software-intensive systems for supporting work activities.

Fairley et al. [FTB94] criticise traditional systems development methods—analyse user needs, develop requirements specification, validate requirements specification, and so on—on two grounds. They claim that traditional methods do not facilitate communication among users, buyers, and developers. They also claim that traditional methods do not emphasise the importance of specifying the operational requirements for the envisioned system. The net effect is that this “leaves the users and buyers uncertain as to whether the requirements specification will provide the needed operational capabilities” [FTB94] ¹.

To tackle these problems, Fairley et al. propose that the process of Concept Analysis, which is described below, should be used to produce a Concept of Operations document. Concept Analysis should be performed as the first stage in the systems development process. The Concept of Operations document should be used to facilitate the derivation of all subsequent development artefacts including in particular the requirements specification. The Concept of Operations document is intended to act as a bridge between the users’ operational requirements and the technical specifications produced during the later stages of system development. Concept Analysis has thirteen stages as shown below:

1. Determine the objectives, roles, and stakeholders
2. Customise and agree the form of the Concept of Operations document
3. • For the current system:
   – Stakeholders describe goals

¹Page 40
CHAPTER 5. RELATING THE SOCIO-TECHNICAL AND
GOAL-ORIENTED APPROACHES

- Stakeholders describe problems
  - For the new (or modified) system:
    - Stakeholders describe goals

4. For the current system:
  - Stakeholders describe boundary
  - Stakeholders describe external system interfaces
  - For the new (or modified) system:
    - Stakeholders describe boundary
    - Stakeholders describe external system interfaces

5. Stakeholders characterise the current system in terms of:
  - operational characteristics
  - operational environment
  - processes
  - modes of operation
  - user classes
  - operational support environment
  - operational maintenance environment

6. For the current system, the stakeholders describe the operational policies and constraints that apply

7. For the new system, the stakeholders describe the operational policies and constraints that apply

8. Characterise the new system in terms of operational characteristics

9. Document operational scenarios for the new system
  - Cover all modes of operation, classes of user, specific operations, and specific processes
  - validate scenarios with users and record information concerning normal operating states and usual conditions relevant to the operation of the new system
  - Design scenarios to support abnormal operations
  - Develop new scenarios for each relevant contingency
  - Repeat theses steps until all operations and their variations are covered.
  - Develop associated test scenarios to validate the operational aspects of the delivered system

10. Validate the new system and the operational scenarios by asking representative users to walk through the test scenarios

11. Stakeholders categorise new system features and operational scenarios as essential, desirable, or optional, and prioritise the items in each category. Stakeholders also describe the omitted features and scenarios.
12. Analyse and describe the implications (impacts) of the new system on each stakeholder group

13. Compare the old and new systems and describe:
   - the benefits of the new system
   - the limitations and disadvantages of the old system

Concept analysis is intended to facilitate the production of a Concept of Operations document. This document provides a logical flow of information as follows:

   - Description of the current system
   - Consideration of, and justification for, needed changes
   - Description of the new system

It should be clear that the Concept Analysis method is similar to the generic model. For example, both emphasise characterising the current system before characterising the new system; for both, important features of the characterisation include goals, and work practices. However, Concept Analysis, at least as it is described in [FTB94], is associated with problems over the level of detail of its presentation. For example, it is not clear how the products of its stages—goals, boundaries, and so on—are to be elicited and expressed. More significantly, it is not clear how the products of one stage are used to facilitate the derivation of products in another stage. For example, it is not clear how goals of the new system are used to derive operational characteristics of the new system. Again, the paper indicates that Concept Analysis is intended to help to surface and resolve conflict among stakeholders over for example goals. But mechanisms for achieving conflict resolution are not proposed. And finally, the paper is silent over the problems of facilitating the co-operation of stakeholders as they enact the Concept Analysis process.

Each of the problems with the two approaches is addressed and either attenuated or eliminated in the new, synthesised approach to goal-oriented requirements engineering that is described in the next chapter.

5.4 Summary

This chapter has proposed a generic model for goal-oriented requirements engineering that integrates the socio-technical approaches and goal-oriented approach reviewed in earlier chapters. Two approaches—ISAC and CACO—considered to be early manifestations of the generic model were also presented and critically reviewed. In the next chapter, the generic model is developed into, and elaborated as, the new, synthesised goal-oriented approach to requirements engineering.
Chapter 6

A synthesised goal-oriented approach
6.1 Introduction

Recent approaches to requirements engineering have partially solved some of the problems with traditional approaches to requirements engineering that are described in Chapter Two, such as neglect of both the wider organisational context and the high-level objectives of an organisation. However, some of these problems remain and other problems have been identified (see Chapters Two and Three). Despite this, some principles, ideas, and techniques of the recent approaches seem to be effective and are therefore considered worth retaining in any new approach. The problem is to know which parts of the recent approaches to retain and how to integrate them effectively into a complete approach. To address all these problems, a framework of principles and features for building requirements engineering methods is derived from a critical review of the established approaches. The framework is used to synthesise a coherent and integrated goal-oriented approach to requirements engineering. This chapter describes this new approach. The retained principles, ideas, and techniques are described in the next section and summarised in a framework of principles and features. Following this, the new synthesised approach to goal-oriented requirements engineering is described. Finally, similar approaches are reviewed in the last section.

6.2 Rationale for the synthesised approach

To address the problems referred to in Chapter Two, a detailed goal-oriented approach to requirements engineering has been synthesised using a number of principles, ideas, and features, which have been drawn from the approaches reviewed in Chapters Two and Three. This section outlines the new approach, indicates the source of the main principles, ideas, and features that it embodies, and attempts to justify the main features of its design. At the end of the section, the general principles and the features of the goal-oriented and process-oriented methods, both of which have informed the design of the new goal-oriented approach, are summarised in a framework. This framework is used in Chapter Eight, Critical Evaluation, to assess the new method, and to compare it to the established methods that have been critically reviewed in earlier chapters.

The most important principle guiding the design of the new approach is that in order to improve some part of the real-world, that part must first be understood and conceptualised. Unless one is going to ignore the whole of some part of the real-world and reconceive it from a “greenfield” position, it is better to understand that part of the real-world, since, in general, it is easier to make incremental, adaptive changes to what is well understood than to what is not well understood. And, although in some cases it might be sensible to sweep away the whole of part of the real-world, in many cases it is not, because of the costs—large-scale redevelopment, training, and so on—and the risk that something completely new will not work. This principle has been adopted in the division of the approach into the following
CHAPTER 6. A SYNTHESISED GOAL-ORIENTED APPROACH

three consecutive parts:

- Conceptualise the current served system
- Investigate IT and benchmarks, and select stakeholders
- Derive requirements for new served and serving system

The first part focuses on understanding the current nature of the area under investigation. The second part selects the stakeholders and optionally investigates benchmarks and IT. And the third part uses the results of the first part to help to improve the area under investigation.

The second most important principle guiding the design is Checkland’s law of conceptualisation: “a system which serves another cannot be defined and modelled until a definition and model of the system served are available” [Che81, WBPB95]. This principle has been adopted in the division of the third part of the approach into two sets of activities that are intended to be performed consecutively: the first set is concerned with conceptualising the new served system, while the second set is concerned with using the conceptualisation in order to derive the requirements for a serving system to support it. In this way the approaches taken by Checkland’s Information Systems Development [WBPB95], Holtzblatt’s and Beyer’s Contextual Inquiry [HB93, HB96, HB98], and by Eason’s socio-technical approach [Eas88] have been emulated.

As with Checkland’s Information Systems Development method and Contextual Inquiry, in the synthesised approach, served systems are conceptualised in terms of goals and work practices (organisational activities). In addition, they are also characterised in terms of their logical structure, as recommended by Eason in his socio-technical approach. So both the current and new served systems are characterised in terms of goals, work practices, and logical structure. Goals are characterised because they constitute the most fundamental feature of a served system; they define its purpose, its reason for existing. Work practices are characterised because they are similarly fundamental in that they facilitate the achievement of goals to a greater or lesser extent. The logical structure is characterised because it is also fundamental. The logical structure expresses the organisation of a served system in terms of its constituent subsystems, the roles associated with them, and the work activities contributed to by the roles.

The features of the design of the new approach that have been discussed to this point are illustrated in figure 6.1.

Part-one of the new approach is also divided into two sets of activities that are intended to be performed consecutively. The first set focuses upon obtaining a broad characterisation of the whole of the served system. The second set focuses upon obtaining a detailed characterisation of just one or a few parts of the served system. The idea here is that since, in general, it would be impossible to improve everything at once, the client should first be made aware of the nature of the broad served system, then be encouraged to reflect upon it, and finally be encouraged to prioritise its constituent parts for subsequent development. A similar idea is embodied in Eason’s socio-technical approach [Eas88], as was noted in Chapter Two. It is expected that this two-part process will facilitate the selection of the most appropriate computer-based systems to develop from all of those that might be
developed in a particular organisational context, and thus help to solve the selection problem which was identified by Checkland [WBPB95]: knowing which information system to build out of all those that might be built.

In part-one of the approach, information about the current served system’s goals, work practices, and logical organisation is elicited from the stakeholders using orthodox requirements engineering elicitation techniques: interviews and questionnaires, for example. In addition, direct users of a focused served system are observed in, and questioned about, both their current work practices and the computer-based systems that currently support them, if any. The idea here is to discover what would and would not be feasible to introduce for direct users in the way of both new work practices and new or modified computer-based systems. This technique is taken from the Contextual Inquiry method [HB93]. Information collected for goals in the new synthesised approach is similar to information for goals collected by standard goal-oriented approaches: for each goal, it includes a goal description, a goal classification (type), a goal owner, a goal priority, and so on. Similarly, goals are modelled using similar techniques to ones used in standard goal-oriented approaches. Information about work practices (that is to say, about organisational activities or processes) is recorded in text descriptions and modelled using the Role Activity Diagram notation [Oul95]. Logical organisation diagrams are drawn using a simple scheme of rectangles to represent logical parts of an organisation, with named roles and named role actors.

In part-two of the approach, stakeholders are selected who will help to enact part-three. In addition, the following phenomena may optionally be investigated: relevant state-of-the art technologies and how similar goals

---

**Figure 6.1:** High-level view of the synthesised approach
have been met elsewhere.

In part-three of the synthesised approach, stakeholders’ goals (new and old) are used to drive the process of generating requirements for computer-based systems. However, unlike some existing goal-oriented approaches such as KAOS [DvF93] and Loucopoulos’ approach [LK95a], in the new approach the stakeholders themselves, facilitated by a requirements engineer, generate incremental and radical ideas (new working practices and new serving systems functionality) for achieving the goals.

In generating such ideas, goal-conflict is likely to occur and one stage of part-three involves the stakeholders in reflecting upon any goal-conflicts in order to resolve them. This mirrors the goal-conflict resolution strategies of traditional goal-oriented approaches.

In addition to facilitating the resolution of identified conflict, the requirements engineer tries to help the stakeholders to appreciate the full impact of their generated ideas. This is done by predicting any additional effects of the generated ideas and then evaluating the effects—positive, neutral, or negative—with respect to each stakeholder. This information is intended to help the stakeholders to select the most appropriate ideas to include in the new served and serving systems that are finally selected.

In the final stages of part-three, models of the stakeholders’ goal hierarchies are used to elaborate the requirements for the serving system, that is to say for computer-based systems that will satisfy the goals of a new served system. These models are used in conjunction with both the resolutions of goal-conflicts and the ideas that have been agreed by the stakeholders for meeting goals.

In addition to the principles, features, and ideas that have just been discussed, four other principles have informed the design of the new approach: maximising the involvement of stakeholders, maximising the use of validation, the need to develop a clearly articulated method, and the need to include in the approach a well-defined syntax and semantics for its models. First, stakeholders are involved as much as possible in all parts of the new approach, both as a source of knowledge about a served system, and as partners with the requirements engineer in developing better served and serving systems. In adopting this principle each of the approaches reviewed in Chapter Two has been emulated. So the new approach should also be able to address problems related to stakeholders exhibited by early requirements engineering approaches: poor domain knowledge, stakeholders rejecting new systems, and so on. These were discussed in Chapter Two. Second, the concept of validation is a key principle of software engineering. Validation involves the requirements engineer checking with the stakeholders that the requirements engineer’s understanding of each of the stakeholders’ contributions—knowledge about current goals, wishes for future goals, knowledge about current problems—is correct. This principle has been adopted and validation steps have been included in the approach wherever possible. Third, in order to encourage the use of this method, it was articulated in terms of both high-level abstractions—parts and stages—and of low-level abstractions—steps and activities. And fourth, an appropriate set of models has been designed to support the method; each of these is presented with a well defined and readily comprehensible syntax and semantics.

93
CHAPTER 6. A SYNTHESISED GOAL-ORIENTED APPROACH

Framework of principles and features for requirements engineering methods

Critical review of the established approaches yielded a number of general principles and features that have been used to inform the design of the new synthesised goal-oriented approach. These principles and features are summarised below as a framework which can be used to inform the design of similar new approaches as well as to assess, and thus to facilitate the comparison of, established approaches. In the framework, the general requirements engineering method principles are as follows. A method should:

- Understand the current broad served system first (conceptualise current first)
- Use an understanding of the broad served system to select a focused served system to improve (broad before focused)
- Conceptualise the focused served system before its associated serving system (served before serving)
- Encourage the proactive, cooperative participation of stakeholders (proactive, cooperative stakeholders)
- Maximise the use of validation (maximise validation)
- Be clearly articulated in terms of a range of levels of abstraction, from relatively abstract—parts and stages, for example—to relatively detailed—steps and activities, for example (clearly articulated method)
- Produce appropriate models, each defined with a clear syntax and semantics (well defined models)
- Encourage in stakeholders an awareness of the state-of-the-art in relevant areas of IT (IT aware stakeholders)

In addition to these general principles, the framework comprises the following features derived from the established approaches that were critically reviewed in earlier chapters:

- Conceptualise goals and problems of the current broad served system (current goals and problems)
- Conceptualise the current work of the broad served system (current work)
- Conceptualise CBS support for current goals and work (current CBS support)
- Identify stakeholders’ new goals for focused served system and resolve conflicting goals (stakeholders’ new goals)
- Design new work to satisfy goals (new work)
- Design alternative CBS solutions to support new goals and work (alternative CBSs)
CHAPTER 6. A SYNTHESISED GOAL-ORIENTED APPROACH

- Predict and assess impact of side effects of alternative CBSs (*impact of side effects*)
- Choose from among alternative CBS solutions and document requirements (*document CBS requirements*)

This framework of principles and features is used in Chapter Eight, Critical Evaluation, to assess and then compare the new synthesised goal-oriented approach and the established approaches.

6.3 The synthesised approach

This section describes the new synthesised approach to requirements engineer. The approach is relevant for a particular class of problems: deriving requirements for computer-based systems that support organisational processes, in other words, work practices. Organisations, including businesses, have goals, from high-level strategic goals, through lower-level process goals, to goals of activities. After discovering these goals, they can be used to derive organisational processes and requirements for computer-based systems that support the processes. Discovering organisational goals and deriving associated requirements is the essence of the approach described here. The approach has three main parts:

- Conceptualise the current served system
- Investigate IT and benchmarks, and select stakeholders
- Derive requirements for new served and serving systems

Each part is described in detail in the following sections.

6.3.1 Part-one: Conceptualise the current served system

Part-one of the synthesised approach comprises four stages, each of which is described in detail below:

- Investigate and model the broad served system
- Investigate and model external views of the served system
- Investigate and model the focused served system
- Summarise investigations and resolve anomalies

**Stage-one: Investigate and model the broad served system:**

Stage-one comprises four steps each of which is described in detail below:

- Identify client
- Elicit and model client’s view of the served system
- Validate client’s view of the served system
- Obtain client’s agreement on focused served system
Step-one: Identify client: The requirements engineer should identify the client. The client is the main person or agency responsible for the domain in which problems are located. The client has the power to authorise the actions of the requirements engineer. The client also has power to make changes to the domain—to a significant amount of its goals or structures or processes—in ways which may affect the problems.

Step-two: Elicit and model client’s view of the served system: The requirements engineer should first elicit the client’s conception of the key features of the served system, that is to say of the domain over which the client has jurisdiction. The key features of the domain are defined in the following list and their interrelationships are illustrated in figure 6.2.

- Structure
- Goals
- Tasks
- Problems
- Impact of problems on goals
- Desired improvements
- Impact of improvements on goals
- Planned major changes

A questionnaire has been designed that is intended to elicit the key features of a domain from a client’s perspective (see Appendix D). Notes of guidance and templates for the additional forms that need to be completed have also been developed (see Appendix D). The requirements engineer administers the questionnaire in order to elicit the client’s conception of the key features of the served system.

After the key features have been elicited, the requirements engineer should model them in a set of diagrams. Each new type of diagram is introduced below in the following way. First, its constituent parts are defined in a UML class diagram. Second, a table is presented that illustrates the symbols that are used to represent each constituent part. Finally, a small example of the diagram type is presented and explained.

The first key feature of a served system is its structure. It is important to capture the structure of the current served system. Unless it is a greenfield site that is being dealt with, any changes will have to be made to the current structure. So the nature of the current structure should be understood; and modelling it will help to achieve this. The structure should be expressed in two types of diagram: a System Map and a System-subsystem Diagram. The System Map is characterised in figures 6.3, 6.4 and 6.5.

The boundary of a client’s domain is shown with a double-bordered rectangle with rounded corners. It is particularly important to depict the boundary because it indicates the area over which the client has jurisdiction. In other words it indicates the area over which the client may legitimately sanction changes to internal structures, goals, processes and tasks.
Figure 6.2: UML class diagram of served system features
The System-subsystem Diagram extends the System Map by showing both the name of the role that manages a subsystem, and the name of the person currently occupying that role. The System-subsystem Diagram is characterised in figures 6.6, 6.7 and 6.8. It records the names of the subsystem managers. Later the requirements engineer will elicit from each of the occupiers of these roles the key features of their subsystem. This will be done to assess the amount of alignment between a client’s perspective of a domain and the controllers’ perspectives of their associated subsystems.

The second key feature of a served system is its goals. The high-level goals of a client’s domain are the most important feature of the domain. They should provide the rationale for all of the activities carried out in the domain. Once a client’s high-level goals have been captured, they may be used to assess the extent to which subsystem behaviour accords with those goals. As an additional use for high-level goals, current goals that survive unchanged into any new system design may also be used as the starting point to derive subgoals. Eventually from these, either requirements for computer-based systems may be derived, or processes may be derived from which requirements for supporting computer-based systems may be derived. The requirements engineer should use the captured high-level goals to create a Client’s High-level Goals Table with the format shown in Table 6.1. The requirements engineer should re-express the data contained in the Client’s High-level Goals Table in a Client’s High-level Goals Diagram. Such a diagram allows the requirements engineer to see at a glance the client’s high-level goals, their perceived importance to the client, their type, and who has assigned them. The Client’s High-level Goals Diagram is characterised in figures 6.9, 6.10 and figure 6.11.

The third key feature of a served system is its tasks. After tabulating and modelling the client’s high-level goals, the requirements engineer should tabulate the client’s view of the tasks associated with the domain. People often find it difficult to state the goals of work that they perform, but find it
### Chapter 6. A Synthesised Goal-Oriented Approach

<table>
<thead>
<tr>
<th>Class</th>
<th>Icon</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subsystem</td>
<td><img src="image1" alt="Subsystem Icon" /></td>
<td>Subsystem of interest shows the boundary of a client's domain</td>
</tr>
<tr>
<td>Association contains</td>
<td><img src="image2" alt="Association Icon" /></td>
<td>Subsystem A contains subsystem B</td>
</tr>
</tbody>
</table>

Figure 6.4: System Map iconography

![System Map Example](image3)

Figure 6.5: System Map example
CHAPTER 6. A SYNTHESISED GOAL-ORIENTED APPROACH

Figure 6.6: System-subsystem Diagram class diagram

Figure 6.7: System-subsystem Diagram iconography

Figure 6.8: System-subsystem Diagram example
Table 6.1: Client’s High-level Goals Table

<table>
<thead>
<tr>
<th>Goal name</th>
<th>Goal description</th>
<th>Goal source</th>
<th>Goal type</th>
<th>Goal priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Support:</td>
<td>To provide computer system support to all users. Provide assistance via the Helpdesk, computer officers and help sheets. Provide a means for users to report problems and for these problems to be investigated and if possible corrected.</td>
<td>Faculty Computing Committee</td>
<td>Achieve Maximise</td>
<td>1</td>
</tr>
</tbody>
</table>

The table’s column headings have the following meanings:

- **Goal name**: A short name for the goal.
- **Goal description**: A natural language description of the goal.
- **Goal source**: The name of the person or agency that assigned the goal. Or “self-assigned” in the case where the client generated the goal.
- **Goal type**: One or more of the following values:
  - Achieve
  - Avoid
  - Maintain
  - Maximise
  - Minimise
  - Increase
  - Decrease
  - Cease from
- **Goal priority**: One or more of the following values:
  - 1 Highest importance
  - 2 Important
  - 3 Not very important
CHAPTER 6. A SYNTHESISED GOAL-ORIENTED APPROACH

Figure 6.9: Client’s goals class diagram

<table>
<thead>
<tr>
<th>Class</th>
<th>Icon</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highest Importance Goal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Important Goal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NotVery Important Goal</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Association</th>
<th>Icon</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>assigns</td>
<td></td>
<td>One agent assigns a goal to another</td>
</tr>
<tr>
<td>achieves, etc.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.10: Client’s goals iconography

Figure 6.11: Client’s goals example
Table 6.2: Domain Tasks Table

<table>
<thead>
<tr>
<th>Task name</th>
<th>Task description</th>
<th>Task priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support Computer</td>
<td>To ensure that all computer systems are maintained in working order, that software</td>
<td>1</td>
</tr>
<tr>
<td>Systems</td>
<td>is upgraded in order to improve the reliability of the systems and to assist in the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>support of the systems.</td>
<td></td>
</tr>
</tbody>
</table>

The table’s column headings have the following meanings:

- **Task name**: A short descriptive name for the task.
- **Task description**: A natural language description of the task.
- **Task priority**: 1, 2, or 3, where 1 is the highest-priority and 3 the lowest.

much easier to describe the tasks that they perform. Since performed tasks achieve goals, there is a direct correspondence between tasks and goals. So the task descriptions are intended to supplement or complement or, in some cases, substitute for goal descriptions. Tasks should be tabulated in a Domain Tasks Table. This table has the format shown in Table 6.2.

The fourth key feature of a served system is its **problems**. After tabulating the tasks, the requirements engineer should tabulate the problems that the client perceives. Each problem describes undesirable states in the client’s domain. For example, in the University Helpdesk case study, which is presented in the next chapter, the client perceived that “logging and tracking where a problem is currently after it has been communicated to the helpdesk” was problematic. Problems are indicators of areas where the behaviour in a domain is likely to be less than optimal. By addressing and resolving problems, it is expected that domain performance will improve. For example, continuing the example described above, by addressing the “logging and tracking” problem, it might be either attenuated or eliminated; and thus domain performance might be made more effective, more efficient, or both. As we shall see later, problems may be readily re-expressed as goals. These in turn may be used to either ultimately derive requirements for computer-based systems, or to derive organisational processes from which requirements for computer-based systems might be derived. So problems may be viewed as important drivers of improvements to a domain, thus it is important to capture and characterise them. The requirements engineer should tabulate the problems expressed by a client in a Domain Problems Table. This table has the format shown in Table 6.3.

The fifth key feature of a served system is the **impact of its problems on its goals**. The questionnaire asks a client to consider which goals are impacted by each of the identified problems. The requirements engineer will review these goals in subsequent work, and try to ascertain whether any of them may be met in new ways which either attenuate or eliminate the problems. The requirements engineer should tabulate the data in a Goals Impacted By Problems Table. This table has the format shown in Table 6.4. The
### Table 6.3: Domain Problems Table

<table>
<thead>
<tr>
<th>Problem name</th>
<th>Problem description</th>
<th>Problem severity</th>
<th>Problem frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem Logging</td>
<td>Logging and tracking where a problem is currently after it has been communicated to the helpdesk.</td>
<td>Severe</td>
<td>Continuous</td>
</tr>
</tbody>
</table>

The table's column headings have the following meanings:

- **Problem name**: A short name for the problem.
- **Problem description**: A natural language description of the problem.
- **Problem severity**: One or more of the following values:
  - Extremely severe
  - Very severe
  - Severe
  - Not very severe
  - Mild
- **Problem frequency**: One or more of the following values:
  - Continuous
  - Hourly
  - Daily
  - Weekly
  - Monthly
  - Yearly
  - Other (as specified)
requirements engineer should re-express the data in the Goals Impacted By Problems Table table in a Goals Impacted By Problems Diagram. The Goals Impacted By Problems Diagram is characterised in figures 6.12, and 6.13.

Table 6.4: Goals Impacted By Problems Table

<table>
<thead>
<tr>
<th>Problem name</th>
<th>Goal name</th>
<th>Goal description</th>
<th>Nature of impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem Logging</td>
<td>User Support</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>User Support / Problem Logging</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The table’s column headings have the following meanings:

- **Problem name**: A short name for the problem. (The problem, and thus its name, should already have appeared in the Domain Problems Table (see Table 6.3)).

- **Impacted goal names**: The names of one or more high-level goals. These should already have been recorded in the Client’s High Level Goals Table (see Table 6.1).

- **Goal description**: A natural language description of the goal.

- **Nature of impact**: A short description of how the problem impacts the achievement by the client of the goal.

The sixth key feature of a served system is the desired improvements associated with it. The questionnaire also invites a client to list the improvements that they would like to make, and to indicate for each one the goals that would be impacted. The information enables the requirements engineer to see the extent to which a client is likely to focus on their most important goals and problems. The improvements should be tabulated in a Desired Improvements Table. This table has the format shown in Table 6.5.

The seventh key feature of a served system is the impact on goals of improvements associated with it. The connections between improvements and goals should be recorded in a Goals Impacted By Improvements Table. This table has the format shown in Table 6.6.

The eighth and final key feature of a served system is the list of major changes planned for it. Major changes that a client has planned for a domain provide an implicit indication to the requirements engineer of the priority that the client has assigned to its goals and problems. These can be checked against corresponding explicit priority indicators, like those in the Client’s High-level Goals table, for example. Any anomalies can be reported back to the client to be addressed and resolved. Additionally, the planned changes indicate to the requirements engineer existing changes in the domain. The requirements engineer needs to be aware of the current status of each planned major change to ensure that subsequent changes fit in with ongoing changes. The requirements engineer should record the planned major changes.

**Step-three: Validate client’s view of the served system**: The purpose of this step is for the requirements engineer to validate his or her under-
CHAPTER 6. A SYNTHESISED GOAL-ORIENTED APPROACH

Figure 6.12: Goals impacted by problems class diagram

Figure 6.13: Goals impacted by problems iconography
Table 6.5: Desired improvements Table

<table>
<thead>
<tr>
<th>Improvement name</th>
<th>Improvement description</th>
<th>Improvement importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster Service</td>
<td>Improve the turnaround of support within the team. Again tied up with logging and tracking of problems. Problems can slip and not get attention as they are not listed anywhere.</td>
<td>Important</td>
</tr>
</tbody>
</table>

The table’s column heading have the following meanings:

- **Improvement name**: A short, descriptive name for an improvement.
- **Improvement description**: A short description of the improvement.
- **Improvement importance**: Indicates the importance of the improvement.
  - Extremely important
  - Very important
  - Important
  - Not very important
  - Unimportant

...standing of the client’s view of the domain context, problems, and potential for improvement. The process for this step is expressed in the following two activities:

- Search for anomalies in the client’s questionnaire data
- Validate questionnaire data with the client

In the first activity, the requirements engineer must examine questionnaire data for anomalies - inconsistencies, ambiguities, lack of clarity, and incompleteness. In the second, the requirements engineer must resolve all anomalies with the client. In this way the requirements engineer checks that their understanding of the client’s view of the broad served system matches the client’s view.

**Step-four: Obtain client’s agreement on focused served system:**

The purpose of this step is to obtain the client’s agreement on goals, problems, and improvements for further investigation. The process for this step is expressed in the following activities:

- Identify the key areas upon which to focus subsequent investigation
- Recommend to the client areas to focus subsequent investigation upon
- Reach agreement with the client on areas to focus subsequent investigation upon
Table 6.6: Goals impacted by improvements

<table>
<thead>
<tr>
<th>Improvement name</th>
<th>Impacted goal name</th>
<th>Goal description</th>
<th>Nature of impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster Service</td>
<td>User support / Problem logging</td>
<td>User support</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Technical investigation</td>
<td></td>
</tr>
</tbody>
</table>

The table’s column headings have the following meanings:

- **Improvement name:** A short name for the improvement. (The improvement, and thus its name, should already have appeared in the Desired Improvements Table (see Table 6.5 on page 107)).
- **Impacted goal names:** The names of one or more high-level goals. These should already have been recorded in the Client’s High Level Goals Table (see Table 6.1 on page 101).
- **Goal description:** A natural language description of the goal.
- **Nature of impact:** A short description of how the improvement impacts the achievement of the goal by the client.

The first activity facilitates the selection of the best computer-based systems to develop from all those that might be developed in a particular organisational context. The key areas upon which to focus subsequent investigation are identified by taking the following actions:

(a) Identify the most important high-level goals

(b) Identify the most important problems

(c) Identify the most important improvements

(d) Identify intersections of (a) with (b) and / or (c)

**Stage-two: Investigate and model external views of the served system:**

The thesis assumes that many organisations are organised in one of three ways: vertically into divisions, departments, and teams; horizontally according to the major organisation processes; or through some combination of both, in a matrix-like way [Har03] \(^1\). Whichever structure exists in an organisation, it is likely that the goals of any level of function or process within an organisation will have been derived from the high-level goals. For example, financial goals will have been translated into more concrete goals for the finance department and then translated again into even more concrete goals for the various groups within that department. Clearly, in practice, the greater the alignment of goals at all levels of a company, the more likely it is that the highest level goals will be satisfied. So it is important to examine the extent of alignment in any area that is being examined. If requirements

\(^1\)Chapter Three
for computer-based systems are going to be derived, it is best that they are requirements for computer-based systems that will contribute to meeting the goals of the domain, which in turn will be consistent with and contribute towards achieving higher-level goals. At this stage in the synthesised approach, the requirements engineer checks the extent of alignment between the client’s perception of their goals and the goals actually assigned to them by external agents or agencies. Any alignment anomalies are subsequently reported back to the client, who is invited to address and resolve them. In order to check the extent of alignment, the requirements engineer should discover which external agents and agencies have legitimately assigned goals to the client for the served system. Then the requirements engineer should discover what goals these external agents and agencies have assigned to the client. For each source of externally assigned goals, an External Goals Table should be constructed. These tables have the format shown in Table 6.7.

Table 6.7: External Goals Table

<table>
<thead>
<tr>
<th>External agents or agency</th>
<th>Goal identifier</th>
<th>Goal description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLAG-1</td>
<td></td>
<td>Provision of help and assistance to users in support of their IT needs.</td>
</tr>
</tbody>
</table>

The column headings in the table have the following meanings.

<table>
<thead>
<tr>
<th>&lt;name&gt;</th>
<th>The name of the external agent or agency, for example Service Level Agreement.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal identifier</td>
<td>A unique identifier for an external goal. It may be made up from an acronym of the agent’s or agency’s name and a number, for example.</td>
</tr>
<tr>
<td>Goal description</td>
<td>A short description of the goal.</td>
</tr>
</tbody>
</table>

When all the external agents, agencies, and their goals have been identified and tabulated, the requirements engineer should construct a mapping matrix that shows the relationship between the client’s perception of their goals and externally assigned goals. This mapping matrix is called the External Goals Client Goals Map. It has the format shown in figure 6.14. The requirements engineer should inspect the matrix for anomalies. For example, a column of blank cells indicates external goals that the client has not identified as high-level goals. A row of blank cells indicates that the client perceives as a high-level goal one that no external agent or agency has assigned. A column with only one or relatively few ticks indicates an external goal that may be recognised only weakly by the client. If discovered, each of these anomalies should be reported to the client, and the client should be encouraged to resolve them.
Stage-three: Investigate and model the focused served system:

This stage comprises six steps each of which is described in detail below:

- Consider the wider served system (optional)
- Elicit and model each subsystem owner’s view of their subsystem
- Validate each owner’s view of their subsystem
- Observe and interview served system workers at work
- Create process models of the served system
- Quantify aspects of the served system (optional)

Step-one: Consider the wider served system (optional): In some cases it may be necessary to consider the wider served system. For example, when the served system is in the business of supplying services, it may be useful to characterise the domain receiving those services. Such a characterisation involves identifying the structure of the domain in terms of its agents and agencies. Once this is completed, the identified agents and agencies may be viewed as a key source of knowledge about the served system. As such they can be asked for their opinions on the problems associated with the served system. They can also be invited to suggest ways in which the served system might be improved and extended.

Step-two: Elicit and model each subsystem owners’ view of their subsystem: In step-two, the requirements engineer should attempt to obtain from each subsystem owner a characterisation of their view of the goals of their subsystem. The characterisation should be in terms of the name, description, assignor name, type, and priority of each constituent goal. This information is needed to assess the degree of alignment between subsystem owners’ goals and the client’s high-level goals. The requirements engineer should report alignment anomalies to the client in stage-four, “Summarise investigations and resolve anomalies”.

The requirements engineer should also try to capture the subsystem owner’s view of the problems identified by the client, and any improvements that they suggest for attenuating or eliminating the problems. This is done
in order to obtain more details about the problems; to elicit potentially useful ideas for attenuating or eliminating them; and to involve in this requirements engineering process people who will be likely to enact any new served and serving systems computer-based systems. The questionnaire in Appendix E may be used by the requirements engineer to obtain the information on goals and problems. The questionnaire contains two standard questions designed to elicit goal information. However, since clients’ problems will vary from project to project, questions designed to elicit subsystem owners’ views on client problems must be designed by the requirements engineer and added to the questionnaire for each new project. The questionnaire may be presented to the subsystem owner as a text document to which they supply textual responses. Alternatively, subsystem owners might be asked the questionnaire questions at an interview conducted by the requirements engineer. The latter option is favoured when the number of subsystem owners is small (one to five), since it allows the requirements engineer to clarify answers and to explore their ramifications “on the spot”. The requirements engineer should re-express a subsystem owner’s questionnaire responses in tabular form. First, for each subsystem owner the requirements engineer should create a Subsystem Owner’s High-level Goals Table. The format of this table is illustrated in Table 6.8. Second, for each of the client’s high-level goals the requirements engineer should create a Client’s Goal Subsystem Owners’ Goals Table that maps relevant subsystem owners’ goals to a client’s goal. The format of this table is illustrated in Table 6.9.

It is important that the requirements engineer should review the set of mapping tables to discover the extent to which the subsystem owners’ goals align with the client’s goals. If the general alignment is not good, then this is a problem that would need to be addressed and resolved in order to improve the effectiveness of the client’s domain. If the specific alignment between the client’s focused served system goals and subsystem owners’ goals was poor, then the requirements engineer knows which subsystem owners must alter their goals in order to improve the alignment.

Finally, the requirements engineer should review the questionnaire responses of the subsystem owners that relate to the specific goals and problems of the focused served system identified by the client. This information may be re-expressed in tables and other figures. If it is possible, the requirements engineer should attempt to identify trends in the subsystem owners’ responses. For example, the requirements engineer may notice that all or most of the subsystem owners behave in the same way with respect to the focused served system, or suggest the same improvements to it.

**Step-three: Validate each owner’s view of their subsystem:** The purpose of this step is for the requirements engineer to validate his or her understanding of each subsystem owner’s view of their goals, of the problems that they have in relation to the key problems identified by the client, and of the improvements that they suggest in relation to these problems. The requirements engineer must examine the questionnaire interview data of each subsystem owner for anomalies such as inconsistencies, ambiguities, lack of clarity and incompleteness. After this, the requirements engineer must take back to each subsystem owner the goal tables produced in the previous stage and any anomalies that have been identified. The requirements engineer
Table 6.8: Subsystem Owner’s High-level Goals Table

<table>
<thead>
<tr>
<th>Subsystem owner</th>
<th>&lt;name&gt; high-level goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal name</td>
<td>Goal description</td>
</tr>
</tbody>
</table>

The table’s column headings have the following meanings:

- **Goal name**: A short name for the goal.
- **Goal description**: A natural language description of the goal.
- **Goal source**: The name of the person or agency that assigned the goal. Or “self-assigned” in the case where the subsystem owner generated the goal.
- **Goal type**: One or more of the following values:
  - Achieve
  - Avoid
  - Maintain
  - Maximise
  - Minimise
  - Increase
  - Decrease
  - Cease from
- **Goal priority**: One or more of the following values:
  - 1 Highest importance
  - 2 Important
  - 3 Not very important

should invite each subsystem owner to validate the requirements engineer’s record of their goals and understanding of the focused served system. The requirements engineer should also try to resolve with each subsystem owner any outstanding anomalies.

**Step-four: Observe and interview served system workers at work:**
During this step the requirements engineer closely observes the way that the work is performed currently in the focused served system. The requirements engineer may also ask questions about the observed work. The purpose of the observation is to obtain a clearer idea both of the detailed nature of the current work, including its current support by computer-based systems, and of the kind of computer-based system features that the workers deem feasible and sensible to introduce. This understanding of current work is necessary because it is from this that new ways of working will have to be introduced. The understanding of what computer-based system elements workers think may and may not be included is important since it is derived from the actual workers’ experiences of performing the work.

The process for this step is enacted for each observation/questioning session; it comprises the following activities:
Table 6.9: Client’s Goal Subsystem Owners’ Goals table

<table>
<thead>
<tr>
<th>Role</th>
<th>Related goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unix System Manager</td>
<td>Keep Unix systems running smoothly.</td>
</tr>
<tr>
<td></td>
<td>Maximise usability of the Unix System.</td>
</tr>
<tr>
<td>Unix Computer Officer</td>
<td>To provide a good, working Unix environment.</td>
</tr>
</tbody>
</table>

The table’s column headings have the following meanings:

- **Client’s goal name**: The short name of one of the client’s high-level goals, for example Achieve System Availability Targets.
- **Role**: The name associated with a subsystem owner’s role.
- **Related goals**: The goals of the subsystem owner which are related to the client’s goal.

- Describe the session’s context
- Observe and question the selected role-holders
- Record problems identified in the session
- Summarise work performed in the session
- Analyse the session’s data
- Review the problems identified in the session
- Record requirements for new work and new computer-based systems

In the first activity, the requirements engineer should describe the context of each observation/questioning session in terms of the following contextual information, which may be informally documented or tabulated: date, time, total elapsed time, location, roles observed, role occupiers’ names, other involved roles, and any other relevant information.

For the second activity, an observation/questioning session comprises a series of events involving the roles being observed. These are interleaved with questions posed by the requirements engineer to the roles in focus. The requirements engineer should record the time of each event and a description of the event. For example, the following extract from a session transcript (see Appendix N) shows the record for three events:

The requirements engineer should also record questions that are put to the observed participants and the answers that they made. The following extract is again taken from the same transcript.

113
I ask K how feasible it would be for her to type in details of problems if a computer-based problem management system were used on the Helpdesk. K says that she would find typing in a lot of details to be too time-consuming. And this would be particularly true when she was busy, for example during lunchtimes.

In the third activity, the requirements engineer should also record any particular problems that characterise an observation session.

In the fourth activity, the requirements engineer should summarise briefly the general nature of the work performed during the session. This is important as it characterises the main features of the current work practices.

In the fifth activity, the requirements engineer should analyse the record of observations, questions, answers, and problems, by categorising each event, totalising the number of events per category, identifying salient features of the numerical data, and reviewing the answers made by observation participants to questions put to them by the requirements engineer. The questions are designed to explore existing and possible future working practices. Answers to the questions may help the requirements engineer to eliminate unfeasible requirements; or to modify requirements they may have been considering either for future working practices or future supporting computer-based systems or both. It is likely that by using this technique the requirements engineer will save the client money that may have been spent on developing features that would not be useful and would therefore probably not be used.

In the sixth activity, the problems identified during an observation session should be reviewed by the requirements engineer. During the review the requirements engineer should imagine ways to attenuate or eliminate these problems in a future focused served system either by the deployment of new working practices, new supporting computer-based systems, or by a combination of both. If existing problems can be attenuated or eliminated in a future system, it will be a more effective system.

In the final activity, the requirements engineer should use the results of the analysis activities to formulate requirements for new working practices (served system) and new supporting computer-based systems (serving systems).

**Step-five: Create process models of the served system:** The purpose of this step is to capture knowledge of the current work, including any IT support, in a form that may be easily validated by the workers, analysed
by the requirements engineer, and used as a medium of communication by both. During the step, the requirements engineer should create models of the processes and any supporting IT that comprise the current work, which has already been determined during previous steps. The models can be expressed using a process modelling language. Process modelling languages support the expression of key features of work, for example roles, activities, interactions, and so on. For example, Ould’s Role Activity Diagrams notation [Oul95] may be used. This notation has been extended by Kawalek et al. to allow Role Activity Diagrams to be annotated with supporting computer-based systems [WKRG99].

**Step-six: Quantify aspects of served system work (optional):** Previous steps may have elicited qualitative or quantitative estimates by served system workers on aspects of their work. The purpose of this step is to obtain more precise quantitative data by such means as observation and counting of important phenomena. Precise quantitative data may be useful in a number of ways. For example, a pattern of activity that occurs frequently seems a better candidate for support by a computer-based system than one that occurs rarely. Or again, if a high volume of data is generated by an activity, then this may suggest that the use of a database rather than a flat file may be appropriate for managing such data. Additionally, quantitative process performance data may also be used to assess whether appropriate resources have been allocated to deploying a process. Such data may also be used to determine if any existing process performance targets are being met. If they are not, then that constitutes another problem that would need to be resolved. If it seems that precise quantitative data will prove useful, then the requirements engineer generates it during this step.

**Stage-four: Summarise investigations and resolve anomalies:**

The results of the investigations carried out in part-one of the approach should be summarised. The summaries should include goal models, organisation structure models, and process models associated with the focused served system. Any outstanding anomalies should be investigated and resolved. Anomalies might include, for example, the case where a goal is assigned by one stakeholder to another, but the assignee does not acknowledge responsibility for the goal.

Additionally, quantified process performance data may be used to assess whether appropriate resources have been allocated to deploying a process. Such data may also be used to determine if any existing process performance targets are being met. If they are not, that constitutes one more problem that would need to be resolved.

### 6.3.2 Part-two: Investigate IT and benchmarks, and select stakeholders

Part-two of the synthesised approach comprises three stages, each of which is described in detail below:  

---

2 Chapter Seven
• Investigate and publish information on relevant IT (optional)
• Benchmarking (optional)
• Assemble stakeholder group

Stage-one: Investigate and publish information on relevant IT (optional):
This stage invites the requirements engineer to optionally investigate the state of the art in areas of IT which may be relevant to supporting organisational processes.

Stage-two: Benchmarking (optional):
This stage invites the requirements engineer to see how similar served systems are served elsewhere, that is to say in other organisations.

Stage-three: Assemble stakeholder group:
In the final stage of part-two, a small group of stakeholders should be assembled by the requirements engineer to help to enact part-three. Following Macaulay [Mac96], the requirements engineer should attempt to identify stakeholders belonging to the following categories:

• Stakeholders with a financial interest in the proposed computer-based system
• Stakeholders who have an interest in the use of the proposed computer-based system. These may be of three kinds:
  – Predicted frequent users of the proposed computer-based system
  – Predicted occasional users of the proposed computer-based system
  – People who it is predicted might be otherwise affected by the proposed computer-based system
• Stakeholders responsible for the design of the proposed computer-based system
• Stakeholders responsible for the development of the proposed computer-based system
• Stakeholders responsible for the deployment, administration, and maintenance of the proposed computer-based system

The requirements engineer should try to ensure that the group represents the different categories and sub-categories of stakeholders that may be involved. At the same time, for practical reasons, the requirements engineer should also try to ensure that the group does not grow too large: around six people is considered a good size.
6.3.3 Part-three: Derive requirements for new served and serving systems

Part-three of the synthesised approach comprises five stages, each of which is described in detail below:

- Model stakeholders’ goals
- Identify new served and serving systems
- Identify goal conflict and outline new served and serving systems
- Resolve goal conflict and refine new served and serving systems
- Document requirements for new served and serving systems

Stage-one: Model stakeholders’ goals:

Stage-one comprises five steps each of which is described in detail below:

- Elicit and validate the client’s view of problems, goals, and constraints for the new served and serving systems
- Review other stakeholders’ views of problems, goals, and constraints
- Transform all problems into goals
- Create goal hierarchy models
- Create composite goal hierarchy model

Step-one: Elicit and validate the client’s view of problems, goals, and constraints for the new served and serving systems: During part-one of the synthesised approach, a client will have identified a focused served system, the area of their domain which they want to improve. Improvements may take any of the following forms:

- Eliminating or attenuating current problems
- More efficient/effective mechanisms to satisfy current goals
- The introduction of new goals and the mechanisms to meet them
- Or some combination of all of these

In this step, the requirements engineer elicits from the client problems specific to the focused served system. In addition, the requirements engineer elicits the client’s goals for new served and serving systems as well as any constraints associated with the development of either. The requirements engineer may use an interview or questionnaire to elicit the information on goals, problems and constraints. The following three requests for information should be put to the client and the answers should be recorded. Can you describe some of the problems that you would like to see eliminated or attenuated in any new served and serving systems? Can you outline some of the goals that you would like to see satisfied or satisficed by new served and serving system? Do you have any specific constraints on the development
of new served and serving systems? Following the interview, the requirements engineer should tabulate the data obtained in three tables: a Client’s Problems Table, a Client’s Goals Table, and a Client’s Constraints Table. The format of each of these tables is shown in Tables 6.10, 6.11, and 6.12 respectively.

Table 6.10: Client’s Problems Table

<table>
<thead>
<tr>
<th>Problem identifier</th>
<th>Short problem description</th>
<th>Problem description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1P4</td>
<td>It is hard to access solutions to infrequently occurring but hard-to-fix problems.</td>
<td></td>
</tr>
</tbody>
</table>

The columns of this table have the following meanings:

- **Problem identifier**: This has the format “CnPm”, where n is a client number, for example one, and m is the number of the problem: numbers are assigned contiguously starting from one.
- **Short problem description**: A short description of the problem.
- **Problem description**: A longer description of the problem.

Next, the requirements engineer should invite the client to validate the elicited goal, problem, and constraint information. A variety of means may be used to achieve this end. For example, in one approach, the requirements engineer may write a report summarising the goal, problem, and constraint data; then send the report to the client for them to review its completeness and accuracy. The requirements engineer would subsequently amend any incompletenesses and inaccuracies identified by the client.

**Step-two: Review other stakeholders’ views of problems, goals, and constraints:** In this step the requirements engineer should elicit other stakeholders’ views of problems, goals and constraints associated with the new served and serving systems. For some stakeholders or classes of stakeholders this information will have been elicited during part-one of the approach. For example, subsystem managers are stakeholders and this information would have been elicited from them already during the step “Elicit and model each subsystem owners view of their subsystem” (part-one, stage-three, step-one). And again, other stakeholders may come from the environment of the served system. The required information would have been elicited from them during steps, “Investigate and model external views of the served system” (part-one, stage-two) and “Consider the wider served system” (part-one, stage-three, step-one). However, for other stakeholders, the goal, problem and constraint data may not yet have been elicited. For each of these stakeholders, the requirements engineer must elicit this information. The requirements engineer may use a questionnaire or inter-
### Table 6.11: Client’s Goals Table

<table>
<thead>
<tr>
<th>Goal identifier</th>
<th>Short goal description</th>
<th>Goal description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1G4</td>
<td></td>
<td>Maintain database of common problem-solution descriptions, and provide easy and efficient access to that information for users.</td>
</tr>
</tbody>
</table>

The columns of this table have the following meanings:

- **Goal identifier**: This has the format “CnGm”, where n is a client number, for example one, and m is the number of the goal: numbers are assigned contiguously starting from one.
- **Short goal description**: A short description of the goal.
- **Goal description**: A longer description of the goal.

### Table 6.12: Client’s Constraints Table

<table>
<thead>
<tr>
<th>Constraint identifier</th>
<th>Short constraint description</th>
<th>Constraint description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1C4</td>
<td></td>
<td>A new system cannot depend upon an increase in personnel or lead to downsizing.</td>
</tr>
</tbody>
</table>

The columns of this table have the following meanings:

- **Constraint identifier**: This has the format “CnCm”, where n is a client number, for example one, and m is the number of the constraint: numbers are assigned contiguously starting from one.
- **Short constraint description**: A short description of the constraint.
- **Constraint description**: A longer description of the constraint.
view, for example, to elicit this information and record the answers to the three questions that were put to the client (see previous step). Whether the information is already available or has to be elicited during this step, the requirements engineer should tabulate the data in a set of tables. For each stakeholder there will be three tables: a Stakeholder Problems Table, a Stakeholder Goals Table, and a Stakeholder Constraints Table. The format of the Stakeholder Goals Table is shown in table 6.13. The format of the other two tables is similar. The problems and constraints elicited from stakeholders are tabulated in similar tables.

Table 6.13: Stakeholder Goals Table

<table>
<thead>
<tr>
<th>Goal identifier</th>
<th>Short goal description</th>
<th>Goal description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3G3</td>
<td>Easy access to ad hoc queries</td>
<td>In a user-problem management system it should be easy to access an ad hoc user-problem query service.</td>
</tr>
</tbody>
</table>

The columns of this table have the following meanings:

- **<Stakeholder name>:** The name of a stakeholder, for example “Stakeholder one”.
- **Goal identifier:** This has the format “SnGm”, where n is a stakeholder number, for example three, and m is the number of the goal—numbers are assigned contiguously starting from one.
- **Short goal description:** A short description of the goal.
- **Goal description:** A longer description of the goal.

**Step-three: Transform all problems into goals:** Problems with various aspects of a focused served system signal areas where the focused served system might be improved. In the synthesised approach, problems are first transformed into goals, and then the goals are used to derive requirements for computer-based systems, either directly or indirectly. A problem may be viewed as having a strong relationship to a corresponding goal. For example, the problem “we are making less than 10% profit per year” is related to the corresponding goal “make more than 10% profit per year”. If the goal is satisfied, the problem is eliminated; if the goal is satisficed, the problem is attenuated. “Analysis usually results in a list of problems...negating these formulations leads to a first list of goals to be achieved by the system-to-be” [VL01]. Usually it is straightforward to see how a problem may be transformed into a corresponding goal whose achievement will eliminate the problem. In the synthesised approach, the requirements engineer should consider in turn each stakeholder problem expressed in each of the Stakeholder Problems Tables. Each problem should be transformed into a corresponding
goal, the achievement of which will eliminate or attenuate the problem. The requirements engineer should record the results of the problem transformation work in a set of Transform Problems To Goals Tables. The format of this kind of table is illustrated in Table 6.14.

Table 6.14: Transform Problems To Goals Tables

<table>
<thead>
<tr>
<th>Problem identifier</th>
<th>Short problem description</th>
<th>Goal identifier</th>
<th>Short goal description</th>
<th>Goal description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1P1</td>
<td>Some problems are lost</td>
<td>C1GP1</td>
<td>No lost user-problems</td>
<td>In any new system the number of lost user-problems should be zero</td>
</tr>
</tbody>
</table>

The columns of this table have the following meanings:

- **Stakeholder name:** The name of a stakeholder that has been elicited in the approach, for example client or stakeholder-one or user-two.
- **Problem identifier:** An identifier that has been used either in the Client Problems Table or one of the Stakeholder Problems Tables, for example C1P1 or S1P2.
- **Short problem description:** A short description of the problem. This description is taken from the Client Problems Table or Stakeholder Problems Tables.
- **Goal identifier:** In general this will have the form LnGPm where L is a character identifying a class of stakeholders, for example, “C” for the client stakeholder. n is a number in a sequence from one upwards. Numbers are allocated contiguously. “GP” stands for goal problem. m is a number from one upwards. So, for example, C1GP3 refers to goal problem number three for the client.
- **Short goal description:** One or two sentences expressing briefly the goal that corresponds to the problem.
- **Goal description:** A longer description of the goal.

**Step-four: Create goal models:** In this step, a Stakeholder Goal Hierarchy Diagram is created systematically for each stakeholder. First the stakeholder’s problem goals are laid out. A characterisation of problem goals using the UML is shown in figure 6.15. The problem goals for a stakeholder are obtained from the associated Transform Problems To Goals Table. They are laid out using the iconography shown in figure 6.16. An example is shown in fig 6.17.

Second, the requirements engineer identifies dependencies between any of the stakeholder’s problem goals, and the first model is redrawn to show them. The iconography for dependencies and an illustrative example are shown in figures 6.18 and 6.19.

Third, the requirements engineer adds the stakeholder’s stakeholder goals
CHAPTER 6. A SYNTHESISED GOAL-ORIENTED APPROACH

Figure 6.15: Characterisation in UML of Problem Goal

Figure 6.16: Iconography of Problem Goal

Figure 6.17: Example of a Problem Goals
Chapter 6. A Synthesised Goal-Oriented Approach

Class | Icon | Meaning
--- | --- | ---
Problem Goal | ![PG icon] | 
Association | ![Icon] | Meaning

| supports the achievement of | | 
| conflicts with | | 

Figure 6.18: Iconography of dependencies between Problem Goals

Figure 6.19: Example of dependencies between Problem Goals
to the model. A characterisation of stakeholder goals using the UML is shown in figure 6.20. The stakeholder goals are obtained from a Stakeholder Goals Table or a Client Goals Table. In some cases, some of a stakeholder’s stakeholder goals may be identical or similar to problem goals. In such cases the requirements engineer chooses the most comprehensive goal for the model. The identifier of the goal not chosen is recorded to facilitate traceability. The iconography and an illustrative example are shown in figures 6.21 and 6.22.

Finally, the requirements engineer again reviews the model and infers any “missing” goals. These are goals that fit logically into the model and are deemed to be useful by the requirements engineer but which have not been explicitly stated by any stakeholder. They could be useful in terms of getting stakeholders to understand better goals that they want achieved; in terms of making explicit what stakeholders think is obvious (and therefore does not need stating); or in terms of providing low-level goals from which requirements for supporting computer-based systems might be derived. The “missing” goals identified by the requirements engineer are called inferred goals. The requirements engineer validates any identified inferred goals with the stakeholders to ensure that they accord with their understanding. Figure 6.23 presents a characterisation of inferred goals using UML. The iconography and an illustrative example are shown in figures 6.24 and 6.25.

After each new goal is added to an emerging goal-hierarchy diagram, the requirements engineer checks to see if any goals conflict with one another. Whenever the requirements engineer identifies conflicting goals, he or she makes the affected stakeholder aware of the situation and invites the stakeholder to resolve the conflict. Until such resolution is obtained, the conflict is depicted on the emerging hierarchy diagram with a “goal conflicts with goal” relationship.

**Step-five: Create composite goal hierarchy model:** When a Stakeholder Goal Hierarchy Diagram has been created for each stakeholder, the requirements engineer should attempt to produce a Composite Goal Hierarchy Diagram from the set of hierarchies. Using the Client Goal Hierarchy Diagram as the base hierarchy, the requirements engineer merges each Stakeholder Goal Hierarchy Diagrams with it, one by one. For each Stakeholder Goal Hierarchy Diagram, this involves adding each of its goals in turn to the emerging Composite Goal Hierarchy Diagram.

As before, in some cases, some of a stakeholder’s goals may be identical or similar to goals already present in the composite goal hierarchy. In such cases, the requirements engineer chooses the most comprehensive goal to retain in the composite hierarchy. To facilitate traceability, the requirements engineer records the identity of the goal not chosen as an attribute of the chosen goal.

Also as before, whenever the goal of a stakeholder conflicts with a goal in the emerging Composite Goal Hierarchy Diagram, the requirements engineer must facilitate the resolution of such conflict. Up until now, any conflict identified has been among the goals belonging to the same stakeholder. To resolve this kind of conflict, the requirements engineer works just with the affected stakeholder. Now, however, it is possible that a number of stakeholders may be associated with a goal that conflicts with a goal in
CHAPTER 6. A SYNTHESISED GOAL-ORIENTED APPROACH

Figure 6.20: Characterisation in UML of Stakeholder Goal

Figure 6.21: Iconography of Stakeholder Goal

Figure 6.22: Example of Stakeholder Goals
Figure 6.23: Characterisation in UML of Inferred Goal

<table>
<thead>
<tr>
<th>Class</th>
<th>Icon</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inferred Goal</td>
<td>RE</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.24: Iconography of Inferred Goal
the composite goal hierarchy. Whenever the requirements engineer discovers a stakeholder’s goal that conflicts with a goal on the emerging Composite Goal Hierarchy Diagram, the requirements engineer should scan the other Stakeholder Goal Hierarchy Diagrams looking for similar goals. Following this, the requirements engineer should bring together the stakeholders on all sides of a conflict in order to facilitate its resolution. The resolution may involve agreement by the stakeholders on a compromise goal; or it may involve a more powerful group of stakeholders retaining their goal in the composite goal hierarchy at the expense of another group’s goal. Until resolution is obtained, conflict should be shown on the emerging composite goal hierarchy using a “goal conflicts with goal” relationship.

Stage-two: Identify new served and serving systems:

The main purpose of this stage is for the requirements engineer first to orient all the stakeholders towards both current served and serving systems and their problems, and towards the goals and constraints identified for any new system; and, second, to encourage them to invent new ways of attenuating and/or eliminating the problems and meeting the new and pre-existing goals.

During this stage the requirements engineer begins by writing a report (Report-1) with the following content:

- A summary of the current served and serving systems
- A summary of the problems with the current served and serving systems
- Lists of stakeholders’ goals and constraints for any new system
- An invitation to the stakeholders to undertake the following activities
  - reflect upon the current served and serving systems and their problems
consider the goals and constraints for any new system

invent new ways of:

* attenuating and/or eliminating the problems
* meeting the new goals

Next, the requirements engineer invites all the stakeholders to a meeting (Stakeholder-Meeting-1) at which the following items are distributed and presented to the stakeholders: the report Report-1, the set of Stakeholder Goal Hierarchy Diagrams, and the Composite Goal Hierarchy Diagram. At the meeting, the requirements engineer invites feedback from the stakeholders on both the report (Report-1) and on the goal-hierarchy diagrams. The requirements engineer also encourages the stakeholders to address and resolve any residual goal conflict featured in the goal hierarchy diagrams. Finally, the requirements engineer invites the stakeholders to e-mail the requirements engineer and/or bring to the next meeting ideas they have had for both attenuating or eliminating problems, and meeting the goals.

Stage-three: Identify goal conflict and outline new served and serving systems:

The main purpose of this stage is for the requirements engineer to help the stakeholders to identify both mechanisms for satisfying goals for which there is a consensus, and mechanisms for satisfying goals where consensus is lacking. During this stage the requirements engineer performs the following steps:

- The requirements engineer considers feedback from the stakeholders that validates Report-1 and/or proposes new served and serving system components
- The requirements engineer facilitates a meeting (Stakeholder-meeting-2) with the stakeholders to consider their suggestions for new served and serving system components
- The requirements engineer writes another, second report (Report-2) that presents the following information:
  - uncontroversial design mechanisms in any proposed served and serving systems components
  - areas of conflicting goals among stakeholders and alternative possible compromise mechanisms
  - summaries of the services identified to date to be provided by computer-based systems

First, prior to Stakeholder-Meeting-2, the requirements engineer may receive feedback from stakeholders on Report-1. The requirements engineer should investigate errors highlighted in the feedback and, if necessary, amend and re-issue the report. In preparation for Stakeholder-Meeting-2, the requirements engineer needs to reflect on feedback about new socio-technical
mechanisms for satisfying the stakeholders’ goals. In particular the requirements engineer should try to calculate the impact of each mechanism on the stakeholders’ goals and predict other significant outcomes (effects) of the mechanisms.

Next, at Stakeholder-Meeting-2, the requirements engineer first prompts the stakeholders again for validation feedback on Report-1. The requirements engineer then invites the stakeholders to express any new ways they have thought of for satisfying one or more of the stakeholders’ goals. The requirements engineer should record the details of any discussions that occur during the meeting, and, after the meeting, write them up and distribute them to the stakeholders as meeting minutes.

And finally, the requirements engineer should reflect upon both the suggestions following Stakeholder-Meeting-1 and the outcomes of Stakeholder-Meeting-2 in order to write Report-2, which outlines alternative designs for satisfying stakeholders’ goals. The main objectives of Report-2 are as follows: to present agreed socio-technical mechanisms that constitute the design of parts of the new served and serving systems; to highlight areas where stakeholders’ goals conflict, and in such cases to present a range of alternative mechanisms with the potential for achieving consensus among the stakeholders; and to orientate the stakeholders towards validating the report’s content, addressing identified conflict, and responding to outstanding questions.

At Stakeholder-Meeting-2 and prior to it, suggestions will have been made by stakeholders for mechanisms to satisfy stakeholders’ goals. Such mechanisms might involve changes to organisational structures, and changes to both patterns of social interaction and patterns of socio-technical interaction. If it is clear that a suggestion would not lead to goal conflict and it is not rejected by the stakeholders, then the requirements engineer may include it as an agreed mechanism in the new served-serving system pairing. In such a case, the requirements engineer will reflect further on the suggestion after the second meeting and then write up the suggestion in Report-2. In the report the requirements engineer will highlight the requirements for computer-based systems or computer-based system services implied by suggestions.

On the other hand, a suggestion may have been made that satisfies the goals of one group of stakeholders while simultaneously undermining the goals of another. When this situation is recognised, the requirements engineer analyses the context of the conflict and proposes a range of mechanisms which are considered to have the potential for uniting the stakeholders. The requirements engineer should also diagram the conflict in order to help the stakeholders to better understand it. In addition to illustrating the conflict and evaluating the impact of a range of alternative mechanisms on affected groups of stakeholders, the requirements engineer also tries to predict other effects of each alternative. The requirements engineer evaluates their impact from the point of view of the affected stakeholder groups. The result of this work is included in Report-2 so that stakeholders can reflect upon any conflict and the likely consequences of mechanisms to resolve it, and start to think about how to resolve it prior to Stakeholder-Meeting-3.

Report-2 will also contain an overview of the services identified to date to be provided by any computer-based systems for the new serving system.
It will end by reminding the stakeholders that they need to validate its content, and address any identified conflict and outstanding issues before the next meeting.

**Stage-four: Resolve goal conflict and refine new served and serving systems:**

The main purpose of this stage is for the requirements engineer to help the stakeholders to resolve any residual goal conflicts. During this stage, the requirements engineer reviews the feedback from the stakeholders on Report-2 and facilitates Stakeholder-Meeting-3. Feedback from the stakeholders on Report-2 may include the following:

- Validation feedback including questions on the accuracy of the report, and requests for information
- Feedback on one or more of any conflicts described in the report
- Answers to any outstanding questions

For validation feedback, the requirements engineer should investigate each point and where necessary modify Report-2. For feedback on conflicts and outstanding questions, the requirements engineer should reflect upon it, and, if it is judged appropriate, introduce it into the Stakeholder-Meeting-3.

At Stakeholder-Meeting-3 the requirements engineer should encourage the stakeholders to take the following actions:

- Address the identified conflicts
- Validate the non-controversial design components
- Address outstanding issues

Encouraging the stakeholders to engage with these areas may be done in a variety of ways. For example, the requirements engineer may prepare material on each conflict. This should then be distributed and reviewed at the meeting. For each conflict the material might include a description and a range of possible resolutions. Again the requirements engineer might prepare a list of validation questions that encourage the stakeholders to think about the details of non-controversial designs and their rationale. This should also be distributed during the meeting and worked through in a systematic manner.

The requirements engineer should update the individual and composite goal hierarchy diagrams to reflect agreements made by the stakeholders that resolve conflict.

**Stage-five: Document requirements for new served and serving systems:**

During this stage, the requirements engineer considers the mechanisms that have been agreed for achieving agreed goals. This agreement is attained at Stakeholder-Meeting-2 and Stakeholder-Meeting-3 and is documented in Report-2 and in the minutes for Stakeholder-Meeting-3. The requirements for computer-based systems (the new serving system) are now derived from
these agreed ways of satisfying the agreed served system goals. First, process models representing the new served system are drawn. These are annotated with the location of serving system computer-based systems. The logical structure of the new served system is also given. In addition, any implementation ideas that have been proposed in earlier stages are documented. Next, the derivation of requirements referred to above is carried out in the following manner. The requirements engineer should first refamiliarise himself or herself with the following material: the agreed non-controversial mechanisms, the agreed resolutions of conflict, and the agreed resolutions of outstanding issues. These are documented in Report-2 and in the minutes for Stakeholder-Meeting-3. After this, the requirements engineer re-examines the Composite Goal Hierarchy Diagram. During the examination, the requirements engineer scrutinises each leaf-goal in turn and attempts to derive requirements for serving system computer-based systems from each. The requirements engineer does this by asking and answering the following question: “what functionality should be provided by a computer-based system in order to satisfy this goal?” The answer to the question (zero, one, or more requirements) is recorded in a table like the one in Table 6.15. Sometimes, one or more questions that are related to a requirement will immediately occur to the requirements engineer. These should be added to the table for later consideration by the stakeholders. Similarly, one or more observations related to a requirement may occur to the requirements engineer. For example, an implementation implication may occur to the requirements engineer upon expressing a requirement. These should also be added to the table. They should be reviewed later on and a decision should be taken on what to do with each one. Finally, after examining the leaf-goals, the requirements engineer should examine goals higher up the hierarchy and process each one in the manner just described.

6.4 Summary

This chapter has presented a new goal-oriented approach to requirements engineering that has been synthesised from the socio-technical and goal-oriented approaches reviewed in earlier chapters. It has also provided a justification for the inclusion and positioning of the main parts of the new approach. The next chapter describes the outcome of applying the new approach to a non-trivial, real-world case study.
Table 6.15: Format of table for stakeholders’ goals and corresponding requirements

<table>
<thead>
<tr>
<th>Goal number</th>
<th>Description</th>
<th>Requirement number</th>
<th>Description</th>
<th>Notes (N) and Questions (Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>REG11</td>
<td>Record details of each user problem and solution</td>
<td>R 1.1</td>
<td>There will be a CBPMS. It will provide an interface to receive details of long-term user-problems</td>
<td>N1: the CBPMS will not support short-term user-problems</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R 1.2</td>
<td>The CBPMS will permanently store details of long-term user-problem until it is deleted</td>
<td>Q1: what should the CBPMS look like?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R 1.3</td>
<td>The details to be stored for each long-term user problem are presented in Report-2 plus “estimated time to completion” and “problem solver list” attributes</td>
<td>N2: the CBPMS will not support printout, borrowing or buying activities</td>
</tr>
</tbody>
</table>
Chapter 7

Case study
7.1 Introduction

Chapter Six describes a synthesised approach to requirements engineering. In order to demonstrate the feasibility and effectiveness of the approach, it was applied to a real-world example: a University Computer Services Department. This chapter first presents a rationale for choosing the case study method. Then the case study is presented.

7.2 Rationale for using a case study

In theory, a number of methods exist for demonstrating the feasibility and effectiveness of approaches like the synthesised approach, including the scientific approach, the case study approach, and the exemplar approach.

Ideally one would like to adopt the scientific approach. Here one could conduct a number of trials of the synthesised method on the same or similar problem context. One could also conduct the same number of trials on the same problem context using other methods whose purpose is the same as the synthesised method. When the results are analysed it would be possible to compare the performance of the synthesised method with the performance of the other methods for the given problem context. This procedure could be repeated with either other similar methods, other problem contexts, or both. Unfortunately the scientific approach is not practicable. The main reason for this is that it would be impossible to find the same problem context over and over again. However, even supposing that this were possible, the scientific approach would be prohibitively time-consuming and expensive.

Two more practical methods of demonstrating the feasibility and effectiveness of the synthesised approach include studying its deployment on a standard requirements engineering exemplar, for example the meeting scheduler exemplar [vLDM95], and studying its deployment on a real-world case study. Finkelstein et al. [FFFvL97] have set out a list of characteristics desired in a requirements engineering exemplar in addition to those it shares with specification exemplars:

- Dealing with a wide variety of aspects, such as goals, costs, resources, performance, responsibilities, priorities, traceability, reliability, evolvability, and so on
- Working with multiple input sources and media such as interviews, observations, system documentation, knowledge about the domain and about similar systems, and so on
- Gathering information from multiple views and stakeholders
- Handling interfering and inconsistent goals giving scope for conflict detection and resolution
- Reasoning about different automation alternatives; reasoning about the interaction/cooperation between the system and its environment
- Reasoning with partial and incoherent descriptions
- Working with large and complex application domains
• Organizing a vast amount of concerns, details and exceptions into a coherent, understandable and manageable structure

• Working in the context of unforeseen changes

It is clear that some of these characteristics might be exhibited through working with a static requirements engineering exemplar. For example, such an exemplar might support “dealing with a wide variety of aspects such as goals, costs, resources, performance, responsibilities, priorities, traceability, reliability, evolvability, and so on.” However, some of the characteristics involve real activities of two or more people: for example, “gathering information from multiple views and stakeholders” and “working in the context of unforeseen changes”. It is difficult to see how such characteristics can be exhibited through working with a static requirements engineering exemplar document.

However, a real-world case study is able to support most of the characteristics listed by Finkelstein et al. and certainly the ones requiring real-world activities. According to Potts [Pot93]: “Case studies are valuable because they incorporate features that problems in the laboratory cannot exhibit.” Although a number of researchers have justified the use of the case study in the context of educational research (see Yin [Yin94] and Merriam [Mer93]), its use has not been similarly justified in the context of software engineering research in general, and requirements engineering research in particular. However, it is clear that in practice the case study is used for a variety of purposes, including evaluation, by the requirements engineering researcher. For example, Potts [Pot93] uses the case study approach to evaluate the suitability of different kinds of object-oriented analysis method for different kinds of system; Sommerville et al. [VS00] use a case study approach to both demonstrate and illustrate the feasibility of an ethnographic approach to requirements engineering; and McDermid et al. [NMB02] use a case study to surface problems with the use case notation and the limitations of the use case creation process in the context of embedded systems. So the case study has been used to illustrate the feasibility of an approach, to evaluate an approach, and to surface problems with an approach. And the case study described below is intended to do the same. In other words, it is intended to show the extent to which the synthesised approach is feasible at all and how effective it is. The case study is also intended to surface problems inherent either in its notation or its process, problems that may be addressed and subsequently resolved. In its problem solving role, the case study is similar to prototyping, where user feedback informs the next development iteration. Here the results of a case study may be used to inform the next iteration of the approach. One of the charges laid against the use of the case study as a research method is that the results of a case study cannot necessarily be generalised. Potts [Pot93] acknowledges the force of this charge and responds that one “should always further validate tentative conclusions”. Yin [Yin94] is more sanguine about the legitimacy of generalising from case studies.
7.3 The case study

The following section describes a case study that was carried out using the synthesised approach. The problem context for the case study was a subgroup of a University Faculty called COTS. The COTS group maintain the various computer systems deployed in the Faculty and also provide a Helpdesk to address technical problems encountered by users. The case study involved the application of the synthesised approach, described in Chapter Six, within the COTS context. The application of the various parts, stages, and steps of the approach are described below. The numbers and names below refer to the numbers and names of the various parts, stages, and steps of the synthesised approach. An overview of the approach is given to remind the reader of its form.

- Conceptualise the current served system
- Investigate IT and benchmarks, and select stakeholders
- Derive requirements for new served and serving systems

7.3.1 Part-one: Conceptualise the current served system

Part-one of the synthesised approach comprises four stages:

- Investigate and model the broad served system
- Investigate and model external views of the served system
- Investigate and model the focused served system
- Summarise the investigations and resolve anomalies

Stage-one: Investigate and model the broad served system:

This stage comprises four steps:

- Identify client
- Elicit and model client’s view of the served system
- Validate the client’s view of the served system
- Obtain the client’s agreement on the focused served system

the enactment of each step is now described in detail.

Step-one: Identify client:  The group within the faculty that supports the computing infrastructure is known as COTS (Computer Operators and Technicians). The COTS manager has both the responsibility and the power to perform the following actions:

- Set goal priorities
- Introduce some kinds of new goals
- Reorganise COTS organisational structures
• Create and recreate COTS procedures
• Suggest that new staff be employed

The COTS manager is able to set goals that are consistent with the goals set for COTS by the Dean. However, the manager cannot introduce new goals that have the same or higher priorities than these. The COTS manager at the start of the case study was Mr Jon Ward. The requirements engineer considered Mr Jon Ward to be the client. The requirements engineer did not consider any of the COTS officers as potential clients because the requirements engineer wanted to consider the whole COTS operation as a source of problems and opportunities, not just a particular part of COTS. Neither did the requirements engineer consider the Dean of the Faculty as the client. Although the Dean has responsibility for COTS, he has delegated its management to a COTS manager.

Step-two: Elicit and model client’s view of the served system: The client was presented with a Client Questionnaire Pro-forma, a set of associated forms, and a set of notes to help complete the questionnaire (see Appendix D). The client was then asked to complete the questionnaire. The client’s completed questionnaire was transcribed by the requirements engineer and is presented in Appendix F. The data from this Appendix has been re-expressed below in the format of the tables and diagrams introduced in the previous chapter.

The previous chapter identified structure as one of the key features of a served system, the domain over which a client has jurisdiction. It is important to capture the current structure of a served system for the following reasons. First, it helps to position a client’s domain within an organisation—the requirements engineer and stakeholders can see whose goals the client should support. Second, it depicts the internal structure of a client’s domain—the requirements engineer and stakeholders can see whose goals should contribute to the client’s goals. Third, the current structure may itself be redesigned in the process of designing new ways of meeting new and current goals, so it needs to be understood. Fourth, and finally, it provides the organisational context in which agents enact processes to meet goals; a context which should be appreciated by the requirements engineer and stakeholders as they undertake their work together. The boundary of the domain of the case study client is illustrated by a System Map (figure 7.1). Its structure is illustrated by a System-subsystem Diagram (figure 7.2).

It is important to capture a client’s high-level goals for the following reasons: first, they provide the rationale for all the processes (work) currently being enacted (performed) in a client’s domain. Thus indirectly they provide the rationale for the computer-based systems that support those processes. And, as new high-level goals are identified and added to the current set, they continue to provide a rationale for retained, redesigned and new processes and, indirectly, for retained, redesigned and new computer-based systems that support the processes. Second, the high-level goals of a client’s domain should support the goals of domains superordinate to the client’s; the client’s goals can be checked to see how well they currently conform to those superordinate goals. Third, and finally, the high-level goals of the managers of the subsystems of a client’s domain should support the achievement of
Figure 7.1: System map showing boundary of client’s domain

Figure 7.2: Sub-divisions in the client’s domain
Table 7.1: Client’s high-level goals

<table>
<thead>
<tr>
<th>Goal name</th>
<th>Goal description</th>
<th>Goal source</th>
<th>Goal type</th>
<th>Goal priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Support: Support users</td>
<td>To provide computer system support to all users. Provide assistance via the help desk, computer officers and help sheets. Provide a means for users to report problems and for these problems to be investigated and if possible corrected.</td>
<td>Faculty Computing Committee</td>
<td>Achieve; Maximise</td>
<td>1</td>
</tr>
<tr>
<td>Computer Service: Achieve systems availability targets</td>
<td>To maintain the computer systems within our responsibility and ensure we meet the availability targets.</td>
<td>Faculty Computing Committee</td>
<td>Achieve</td>
<td>2</td>
</tr>
<tr>
<td>Technical Support: Contribute to Faculty IT strategy</td>
<td>To offer technical support, in the form of advice, to the faculty in order that the faculty can make strategic decisions.</td>
<td>Client’s manager (the Dean)</td>
<td>Increase</td>
<td>2</td>
</tr>
<tr>
<td>Technical Investigation: Deploy best kit</td>
<td>To investigate new technologies and systems for possible implementation within the faculty.</td>
<td>Self-generated</td>
<td>Increase</td>
<td>2</td>
</tr>
<tr>
<td>Training: Support staff development</td>
<td>To encourage the computer officers to develop their skills and knowledge.</td>
<td>Self-generated</td>
<td>Increase</td>
<td>2</td>
</tr>
<tr>
<td>Liaison: Contribute to University IT strategy</td>
<td>To liaise with Information Technology Services to help to define the strategy for computer services for the university.</td>
<td>Self-generated</td>
<td>Maintain</td>
<td>3</td>
</tr>
</tbody>
</table>

The last chapter pointed out that some people find it easier to articulate the tasks that they perform than the goals that they are trying to achieve. Thus it is important to elicit information on tasks from the client. The main domain tasks identified by the client are tabulated in a Domain Tasks Table (Table 7.3). If the information in the Domain Tasks Table is compared with...
the information in the Client’s High-level Goals Table, it seems generally that there is a good correspondence between the high-level goals and the tasks. In this way the elicited information about tasks has confirmed the appropriateness of the information elicited about high-level goals. However it can also be seen that two possible new current high-level goal have been identified: viz to sell discs, manuals, and some teaching materials, as well as to maintain and support the laboratories. So the task information elicited here has also supplemented the information on high-level goals.

Problems perceived in a domain indicate areas where the domain behaviour is likely to be non-optimal. So problems may be viewed as important sources of, and thus drivers of, improvements to a domain. By addressing and resolving problems, it is expected that domain performance will improve. Thus it is important to characterise domain problems in order to resolve them. As the last chapter indicated, resolution involves transforming the problems to goals from which requirements for new processes and supporting associated computer-based systems may be derived. The Domain Problems Table (Table 7.4) records the client’s view of the main problems in his domain of responsibility.

Generally, the problems which should be addressed soonest are those which impact the most important high-level goals. This information can be identified easily in a Goals Impacted By Problems Table and its corresponding diagram. The Goals Impacted By Problems Table (Table 7.5), records the case study client’s view of the goals that are impacted by the problems.
### Table 7.2: High-level goals assigned to sub-division managers

<table>
<thead>
<tr>
<th>Sub-division name</th>
<th>Goal</th>
<th>Description</th>
<th>Type</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help Desk</td>
<td>User Support / Problem Logging</td>
<td>To provide support to users in terms of offering help sheets, manuals and minimal technical advice. Also provide a facility for users to log faults. Also provide a means of communicating the problems to the support teams and to monitor their progression to keep users informed.</td>
<td>Increase</td>
<td>1</td>
</tr>
</tbody>
</table>

![Figure 7.4: High-level goals assigned to sub-division manager]

Figure 7.4: High-level goals assigned to sub-division manager
### Table 7.3: Domain tasks identified by the client

<table>
<thead>
<tr>
<th>Task name</th>
<th>Task description</th>
<th>Task priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support Computer Systems</td>
<td>To ensure that all computer systems are maintained in working order, that software is upgraded in order to improve the reliability of the systems and to assist in the support of the systems.</td>
<td>1</td>
</tr>
<tr>
<td>User Support</td>
<td>Offer technical support to all users facing difficulties. This is achieved by providing help sheets and offering one-to-one sessions to help with the problem.</td>
<td>1</td>
</tr>
<tr>
<td>Package Support</td>
<td>Where possible offer technical support for some of the installed packages.</td>
<td>2</td>
</tr>
<tr>
<td>Print Service</td>
<td>Offer a service where users can print laser printouts which can then be collected from the helpdesk.</td>
<td>2</td>
</tr>
<tr>
<td>Problem Reporting</td>
<td>Allow problems to be reported to the help desk which can then be allocated to the relevant system support for remedy.</td>
<td>1</td>
</tr>
<tr>
<td>Technical Report</td>
<td>Offer technical support to the faculty so that strategic decisions can be made.</td>
<td>2</td>
</tr>
<tr>
<td>Investigate New Technology</td>
<td>Actively look at new technology both hardware and software with a view of its use within the faculty.</td>
<td>1</td>
</tr>
<tr>
<td>Installation</td>
<td>Install new hardware/software systems within the faculty and configure for use.</td>
<td>1</td>
</tr>
<tr>
<td>Laboratory Support</td>
<td>Maintain and support the labs to ensure that they are fully functional. Try and maintain an environment which is pleasant to work in.</td>
<td>2</td>
</tr>
<tr>
<td>Selling Consumables and Teaching Materials</td>
<td>Sell disks and manuals and some teaching materials.</td>
<td>2</td>
</tr>
</tbody>
</table>

### Table 7.4: Domain problems identified by the client

<table>
<thead>
<tr>
<th>Problem name</th>
<th>Problem description</th>
<th>Problem severity</th>
<th>Problem frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem Logging</td>
<td>Logging and tracking where a problem is currently after it has been communicated to the helpdesk.</td>
<td>Severe</td>
<td>Continuous</td>
</tr>
<tr>
<td>User Focus</td>
<td>Not focusing on the customer as the main reason we are here.</td>
<td>Severe</td>
<td>Continuous</td>
</tr>
</tbody>
</table>
Table 7.5: Goals impacted by problems

<table>
<thead>
<tr>
<th>Problem name</th>
<th>Impacted goal</th>
<th>Goal description</th>
<th>Nature of impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem logging</td>
<td>User Support</td>
<td>No response.</td>
<td>No response.</td>
</tr>
<tr>
<td></td>
<td>User Support / Problem Logging</td>
<td>No response.</td>
<td>No response.</td>
</tr>
<tr>
<td>User focus</td>
<td>User Support</td>
<td>No response.</td>
<td>No response.</td>
</tr>
<tr>
<td></td>
<td>User Support / Problem Logging</td>
<td>No response.</td>
<td>No response.</td>
</tr>
<tr>
<td></td>
<td>Technical support</td>
<td>No response.</td>
<td>No response.</td>
</tr>
</tbody>
</table>

This information is modelled in a Goals Impacted By Problems Diagram (figure 7.5). The table and diagram indicate that both of the problems that the client has identified—inadequate logging and tracking of user-problems, and insufficient user focus—impact the client’s most important high-level goal: User Support. So the User Support goal is now a strong candidate for helping the client and requirements engineer to identify the focused served system, and these two problems need to be eliminated or attenuated in the behaviour of any future served system.

It is important to elicit a client’s view of the improvements to a domain that the client considers to be desirable. The requirements engineer and client can together relate this information to the information on high-level goals and problems already tabulated. For example, the client may suggest interesting ways of eliminating and/or attenuating a key problem, ways which can be developed by the requirements engineer and stakeholders in subsequent work. The Desired Improvements Table (Table 7.6) records the case study client’s view of desirable improvements to his domain.

It is important for a client and requirements engineer to appreciate the impact of improvements on goals, in other words to appreciate which goals are associated with each desired improvement. If a suggested improvement
Table 7.6: Improvements to the domain identified as desirable by the client

<table>
<thead>
<tr>
<th>Improvement name</th>
<th>Improvement description</th>
<th>Improvement importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster Service</td>
<td>Improve the turnaround of support within the team. Again tied up with logging and tracking of problems. Problems can slip and not get attention as they are not listed anywhere.</td>
<td>Important</td>
</tr>
<tr>
<td>Liaise With Academics</td>
<td>Improve the relationship with teaching staff so they know what we are doing and we know what they are doing.</td>
<td>Important</td>
</tr>
</tbody>
</table>

is fully developed, then it will be expected to have a measurable impact on these goals and problems. For example, the goal may be achieved more effectively or efficiently and/or any associated problems may be attenuated or eliminated. The Goals Impacted By Improvements Table (Table 7.7) records the case study client’s view of goals that would be impacted by his improvements.

Table 7.7: Goals impacted by improvements

<table>
<thead>
<tr>
<th>Improvement name</th>
<th>Impacted goal</th>
<th>Goal description</th>
<th>Nature of impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster service</td>
<td>User Support / Problem Logging</td>
<td>No response.</td>
<td>No response.</td>
</tr>
<tr>
<td></td>
<td>User Support</td>
<td>No response.</td>
<td>No response.</td>
</tr>
<tr>
<td></td>
<td>Technical Investigation</td>
<td>No response.</td>
<td>No response.</td>
</tr>
<tr>
<td>Liaise With Academics</td>
<td>User Support / Problem Logging</td>
<td>No response.</td>
<td>No response.</td>
</tr>
<tr>
<td></td>
<td>User Support</td>
<td>No response.</td>
<td>No response.</td>
</tr>
<tr>
<td></td>
<td>Technical Investigation</td>
<td>No response.</td>
<td>No response.</td>
</tr>
</tbody>
</table>

The planned major changes implicitly indicate to a requirements engineer a client’s view of their most important high-level goals and problems. The changes can indicate seemingly non-optimal choices on the part of a client. The case study client’s planned major changes are expressed in the following statement: “Restructure teams into a User Support Oriented structure. Implement a problem logging and tracking system.” So in this case study, the client’s planned major changes are consistent with the most important identified high-level goal, User Support, and one of its associated problems, Problem Logging.

Step-three: Validate the client’s view of the served system: The full details of the client’s validation of the current served system are con-
tained in Appendix G. The client’s feedback includes clarifications, elaborations, additional information, and confirmations.

**Step-four: Obtain the client’s agreement on the focused served system:** In step’s first activity, “Identify key areas to focus subsequent investigation upon”, the requirements engineer derived the following information from the case study client’s data:

- The client’s *most important goal* is “User Support”: “To provide computer support to all users. To provide assistance via the helpdesk, computer officers, and help sheets. Provide a means for users to report problems and for these problems to be investigated and if possible corrected.”

- The client’s *most important problem* is “Problem Logging”: “Logging and tracking where a problem is currently after it has been communicated to the help desk.”

- The client’s *most important improvement* is “Faster Service”: “Improve turnaround of support within the team. Again tied up with logging and tracking problems. Problems can slip and not get attention if they are not listed elsewhere.”

This information helped the requirements engineer and case study client to select the focused served system as the management of user-problems.

**Stage-two: Investigate and model external views of the served system:**

It is important to check that the high-level goals of a client’s domain align closely with the goals of related superordinate domains. If they are closely related, and they are achieved, their achievement will contribute to the achievement of higher-level organisational goals. Conversely, if they are not closely related, then, whether or not they are achieved, it is improbable that higher-level goals will be supported. The client identified the Dean of the Faculty and the Chair of the Faculty Computing Committee (FCC) as the external sources of his goals. A generic letter requesting confirmation of this was instantiated for each and sent to both sources. A subsequent e-mail from the Dean indicated that a number of goals assigned to the client were recorded in the Computer Officer’s Service Level Agreement document. These goals had previously been agreed by the Dean, the Faculty Executive (including the Chair of the Faculty Computing Committee), and the previous Principal Computing Officer (PCO)—the client’s job title. The same e-mail also indicated the existence of an additional goal assigned to the client by the Dean: to ensure that two new Open Access Laboratories were up and running by a specified date. In a subsequent interview, the Dean provided the requirements engineer with the official job description for the PCO role. This contained a number of additional goals for the client. The Dean also communicated two more goals that he had assigned to the client: contributing to the Faculty Planning Document and enhancing information flows within the faculty.
Table 7.8: External Goals Table for the Dean

<table>
<thead>
<tr>
<th>Goal identifier</th>
<th>Goal description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EG1</td>
<td>Ensure that the new Open Access Laboratories are up and running.</td>
</tr>
<tr>
<td>EG2</td>
<td>Contribute to the Faculty Planning Document.</td>
</tr>
<tr>
<td>EG3</td>
<td>Enhance information flow within the faculty.</td>
</tr>
</tbody>
</table>

Table 7.9: External Goals Table for the Service Level Agreement

<table>
<thead>
<tr>
<th>Goal identifier</th>
<th>Goal description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLAG-1</td>
<td>Provision of help and assistance to users in support of their IT needs</td>
</tr>
<tr>
<td>SLAG-1.1</td>
<td>Main activity will be to deal with ad hoc queries and progression of fault reports.</td>
</tr>
<tr>
<td>SLAG-1.1.1</td>
<td>Aim will be to deal with as much as possible on the spot.</td>
</tr>
<tr>
<td>SLAG-1.1.2</td>
<td>User support retains responsibility for managing the process of resolving problems and keeping users informed of progress.</td>
</tr>
<tr>
<td>SLAG-1.1.2.1</td>
<td>Some matters will need to be referred to appropriate system management teams.</td>
</tr>
<tr>
<td>SLAG-2</td>
<td>Maintenance of the hardware and software required to provide the computing service.</td>
</tr>
</tbody>
</table>

The three External Goals tables, Table 7.8, Table 7.9, and Table 7.10, summarise the goals assigned to the client by external agents and agencies.

The External Goals Client Goals Map (figure 7.6) shows the mapping between the client’s goals and the goals assigned to the client by external agencies. From figure 7.6, it can be seen that all of the client’s high-level goals (first column), except the goal “Liaison: Contribute To University IT Strategy”, are consistent with external goals expressed either in the Service Level Agreement or in the client’s official job description. In particular, it is worth noting that the goal that the client views as his most important, “User Support”, is the subject of a document produced by an external agency: the Service Level Agreement. However, it should also be noted that JDG-2 (communicating staff requirements), JDG-5 (acting as primary interface channel), and JDG-6 (coordinating networking policy) did not feature in the client’s list of his high-level goals. Either he was unaware of these goals or he considered them too unimportant to mention. In addition, the client did not state explicit goals corresponding to goals assigned to him by the Dean concerning new Open Access Laboratories, the Faculty Planning Document, and information flows within the faculty.

In conclusion, it seems that the client’s view of his major high-level goal
Table 7.10: External Goals Table for the Principal Computing Officer Job Description

<table>
<thead>
<tr>
<th>External agents or agency: Principal Computing Officer Job Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goal identifier</strong></td>
</tr>
<tr>
<td>JDG-1</td>
</tr>
<tr>
<td>JDG-2</td>
</tr>
<tr>
<td>JDG-3</td>
</tr>
<tr>
<td>JDG-4</td>
</tr>
<tr>
<td>JDG-5</td>
</tr>
<tr>
<td>JDG-6</td>
</tr>
<tr>
<td>JDG-7</td>
</tr>
</tbody>
</table>

Figure 7.6: External Goals Client Goals Map

is consistent with the view of external agencies that assign goals to him. There is also a good match for most of the client’s less important high-level goals. But there is no match for one of the client’s less important goals and no match for the Dean’s three goals. The requirements engineer noted these potential problems in order to put them to the client in stage-four “Summarise Investigations and Resolve Anomalies”.

Stage-three: Investigate and model the focused served system:

This stage comprises six steps:

- Consider the wider served system (optional)
- Elicit and model each subsystem owner’s view of their subsystem
- Validate each owner’s view of their subsystem
- Observe and interview served system workers at work
The enactment of each step is now described in detail.

**Step-one: Consider the wider served system (optional):** In this case, study of the wider served system was taken to be relevant since the domain under consideration, COTS, was a service provider. The wider domain was taken to include academic and administrative staff and students, since these were the users who experienced the service provided by COTS. So the views of these users on domain problems and potential improvements were elicited, and used to help to confirm the choice of focused served system.

In order to assess the attitude of the faculty’s users towards the services provided by the COTS team, a user questionnaire (see Appendix I) was sent to the members of the faculty (1,656 recipients)—staff and students. The responses were analysed and written up in a report, “Analysis and review of users’ attitude towards, and perceived problems with, COTS services” (see Appendix J). This report was made available to the client shortly after the investigation of the users in the wider served system was complete. The following observations relate to the results of the analysis which are presented in Tables J.1 and J.2 of Appendix J (see page 290). First, approximately 90% of the 89 users who completed the questionnaire, had used COTS services - contacting either the Helpdesk or a Computer Officer - between one and ten times during the period November 1994 to November 1996. 40% of these had used COTS services more than ten times within that period. In other words, COTS services were used widely and frequently. Second, in general, most respondents seemed positive about the quality of the various services received. For example, approximately two thirds indicated that they were either “happy” or “very happy” with the quality of the overall COTS service. Third, while respondents seemed positive about asking both Computer Officers (COs) and Helpdesk personnel for advice or information, after reporting problems they tended to be happier with the quality of service received from Computer Officers (75% vs. 25% respectively either “very happy” or “happy”). Of course, this is the kind of result one would expect given that visiting a Computer Officer generally means visiting someone with expertise in a given area, whereas interacting with Helpdesk staff generally means interacting with students who often lack deep knowledge in a wide range of areas. Fourth, although most respondents were “neutral” towards borrowing manuals and buying materials from the Helpdesk (55% and 65% respectively), this may have been because few of them had actually used either of these services, perhaps being unaware of their existence. Very few respondents indicated that they were “very unhappy” or “unhappy” with either service (5% and 10% respectively). Finally, although 60% were positive about collecting printed output, nearly one fifth were either “unhappy” or “very unhappy” with this service. Most of these dissatisfied respondents were year-two students.

Using these data as indicators of areas where improvements might be contemplated, the quality of the service provided by Helpdesk personnel...
CHAPTER 7. CASE STUDY

after users report problems and the distribution of printouts were both considered to make sensible candidates for areas to review.

The user questionnaire also invited staff and students to do the following:

- Identify problems they had with the provision of services by COTS
  (users’ problems)
- Suggest ways of improving the existing services (suggested improvements)
- Propose new services (suggested new services)

By examining the content of the responses in each of these three areas, the requirements engineer was able to create a set of meaningful categories in such a way that it was possible to place each response in at least one category (see Tables J.6 to J.9 in Appendix J). The set includes categories such as “problem management”, “service quality”, and “printouts”, for example.

For “users’ problems”, Table J.3 of Appendix J (page 290) ranks the categories in descending order of number of occurrences of problems in each category. So, for example, the category “service quality” contained the most occurrences, the category “problem management” the next highest number, and so on. For “suggested improvements”, Table J.4 ranks the categories similarly, and for “suggested new services”, Table J.5 ranks the categories similarly.

Table J.6 shows which “users’ problems”, “suggested improvements” and “suggested new services” responses the requirements engineer identified as belonging to the “service quality” category. Tables J.7, J.8 and J.9 are similarly organised for responses in the categories of “problem management”, “communication” and “printouts”. (Responses in the category of “sales” and “opening times” have been omitted due their “distance” from solutions requiring computer-based systems support.)

As Table J.3 of Appendix J indicates, the “service quality” category contained the most “problem” responses (fifteen) identified by the COTS service users. The “problem management” category contained the second highest number of responses (nine). While the “printouts” and “sales” categories contained seven and six responses respectively, none of the eighteen remaining categories contained more than four responses.

These results seem to provide some support for focusing subsequent work on the “service quality” problems rather than the “problem management” problems. However, there are two reasons why the requirements engineer considered that this would be the wrong focus. First, Table J.6 of Appendix J indicates that most of the “users’ problems” concern the alleged attitude of some Helpdesk personnel towards users. But it is difficult to see how any computer-based system could help to improve this attitude. Second, only one of the respondents provided comments that could be categorised as “service quality” responses. This figure is considered to be too low to indicate that the users had a major problem in this area.

(It is worth noting that Tables J.6 through J.9 of Appendix J contain many suggestions for both improvements to existing services and for new services: for example, improvement number nine in Table J.7, i.e. “Provide Unix and PC support whenever the laboratories are open”.)
In conclusion, the results of the user questionnaire seem to indicate that users are reasonably happy with the current provision of services by the COTS group and do not experience any major problems in common. Of the problems they have, the one that the requirements engineer considers is most likely to lead to an improvement in quality of service provision through support by a computer-based system is the problem of recording and tracking problems reported by users. This problem is associated with the focused served system already agreed by the client and requirements engineer.

The summary of questionnaire results and the analysis of the questionnaire, discussed above, were written up as a report (see Appendix J) and presented to the client. The client’s response to the report was obtained in an interview with him conducted by the requirements engineer. The requirements engineer posed a number of questions and the client’s answers were noted. After the interview, the client’s answers—in note form—were written up by the requirements engineer (see Appendix K). Overall the client thought that the report was good. He was particularly interested in the problems that the users had, and in their suggestions for improvements to existing services and for new services. Although the client felt that some of the users’ suggestions would be difficult to achieve, for example increasing the number of Computer Officers, others were considered relatively easy to achieve.

Step-two: Elicit and model each subsystem owners’ view of their subsystem: For each subsystem in the client’s domain, the requirements engineer determined both the subsystem owner’s view of their goals, and the way that they manage goals, problems, and improvements associated with the focused served system agreed by the client and requirements engineer. The former information is needed to establish the degree of alignment between the client’s high-level goals and the subsystem owners’ goals. The latter information is needed both because it contains greater detail about focused served system problems, and because it may contain useful ideas for attenuating or eliminating them.

In order to elicit the required information from the subsystem owners, the Generic Questionnaire for Subsystem Owners (see Appendix E) was modified. Two new questionnaires were produced (see Appendix E): one to be administered to the Helpdesk Manager and one to be administered to each of the other Computer Officers. Each questionnaire was administered by the requirements engineer in a small, quiet room. The requirements engineer asked the questionnaire questions and noted down the answers. If the respondent’s reply was not immediately understood, further questions were asked until the answer became clear. The complete responses of each Computer Officer to the appropriate questionnaire are shown in Appendix M. These responses were re-expressed in tabular form. An example of one subsystem owner’s goals is presented in the Subsystem Owner’s High-level Goals Table for the Unix System Manager (Table 7.11).

The requirements engineer reviewed the goals in the set of Subsystem Owner’s High-level Goals Tables to see how they related to the goals ex-
pressed by the client (see Table 7.1, on page 139). The results of the review are expressed in the series of Client’s Goal Subsystem Owners’ Goals Tables in Tables 7.12 through 7.14. The review produced a number of key observations. First, for the client’s highest priority goal, Support Users, most of the subsystem owners had expressed ownership of one or more goals that were deemed by the requirements engineer to be supporting goals. And no subsystem owner had expressed a conflicting goal (see Table 7.12). The same result was obtained for the client’s goals Achieve System Availability Targets and Deploy BestKit (see Tables 7.13, and 7.14). Second, no subsystem owner had expressed goals that the requirements engineer could deem to be related to the client’s goal Contribute To University IT Strategy. This is not unexpected since the boundary of the University is four boundaries away from their own boundary of responsibility: first comes their own boundary, (for example the Unix subsystem), then the COTS boundary, then the faculty boundary, then the University boundary. Third, only one goal was expressed that related to the client’s goal Contribute To Faculty IT Strategy: the Unix System Manager expressed ownership of the supporting goal “Justify the presence of unix”. Fourth, only one goal was expressed that related to the client’s goal Support Staff Development: the Unix Computer Officer expressed ownership of the goal “Learn as much as possible about programming languages”. And fifth, and finally, two goals were expressed that were unrelated to the client’s goals. The PC Computer Officer expressed ownership of the goal “Act as stand in systems manager when required”; and the PC Computer Officer expressed ownership of the goal “Promoting the working together of computer officers”, Academics, Users And Managers.

In conclusion, the main finding concerning goals from the questionnaire responses was that the subsystem owners had goals that supported the

<table>
<thead>
<tr>
<th>Goal name</th>
<th>Goal description</th>
<th>Goal source</th>
<th>Goal type</th>
<th>Goal priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unix systems</td>
<td>Keep Unix systems running smoothly</td>
<td>Job description</td>
<td>Maintain</td>
<td>Highest Importance</td>
</tr>
<tr>
<td>User support</td>
<td>Provide user support (troubleshooting)</td>
<td>Job description</td>
<td>Decrease or Avoid</td>
<td>Highest Importance</td>
</tr>
<tr>
<td>User education</td>
<td>Provision of information and education on Unix matters</td>
<td>Self-assigned</td>
<td>Maintain</td>
<td>Not Highest-importance</td>
</tr>
<tr>
<td>Forecast needs</td>
<td>To forecast and meet future Unix-related need</td>
<td>Job description</td>
<td>Maintain</td>
<td>Important</td>
</tr>
<tr>
<td>Maximise usability</td>
<td>Maximise usability of the Unix system</td>
<td>Self-assigned</td>
<td>Maximise</td>
<td>Important</td>
</tr>
<tr>
<td>Justify Unix</td>
<td>Justify presence of the Unix system</td>
<td>Self-assigned</td>
<td>Maintain</td>
<td>Important</td>
</tr>
</tbody>
</table>
client’s goals, including his highest priority goal viz Support Users. Also, none of the subsystem owner’s goals conflicted with any of the client’s goals. In other words, a high degree of alignment was found between the client’s high-level goals and the subsystem owners’ goals.

Table 7.12: Subsystem managers’ goals related to client’s goal Support Users

<table>
<thead>
<tr>
<th>Role</th>
<th>Related goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unix System Manager</td>
<td>Provide user support.</td>
</tr>
<tr>
<td></td>
<td>Provision of information and education on Unix matters.</td>
</tr>
<tr>
<td>Unix Computer Officer</td>
<td>To educate users to use Unix systems efficiently.</td>
</tr>
<tr>
<td>PC System Manager</td>
<td>Provide PC-related support for students and staff.</td>
</tr>
<tr>
<td>PC Computer officer</td>
<td>No goals stated.</td>
</tr>
<tr>
<td>Oracle System Manager</td>
<td>No goals stated.</td>
</tr>
<tr>
<td>Small Systems Systems Manager</td>
<td>No goals stated.</td>
</tr>
<tr>
<td>Helpdesk Systems Manager</td>
<td>Satisfy staff and users.</td>
</tr>
<tr>
<td></td>
<td>Maintain a library for staff and students.</td>
</tr>
<tr>
<td></td>
<td>Minimise cost of student laser-printouts.</td>
</tr>
<tr>
<td></td>
<td>Develop expertise in solving user-problems.</td>
</tr>
<tr>
<td></td>
<td>Maintain a positive attitude towards helping.</td>
</tr>
<tr>
<td></td>
<td>Help foreign students.</td>
</tr>
<tr>
<td></td>
<td>Dispense advice to users on problem solving.</td>
</tr>
<tr>
<td></td>
<td>Identify sources of help for users.</td>
</tr>
<tr>
<td></td>
<td>Respond to e-mailed student-problems within one day.</td>
</tr>
<tr>
<td></td>
<td>Present positive image of Helpdesk group.</td>
</tr>
</tbody>
</table>

After comparing the subsystem owners’ goals to the client’s goals, the requirements engineer next reviewed the information on the current focused served system (user-problem management) that was expressed in the subsystem owners’ responses to the questionnaire. Table 7.15 shows that all the sub-system owners think that they solve some user-problems on-the-spot when they are received. Most sub-system owners estimate that 75% to 90% of user-problems received are solved on the spot. The Unix Systems Manager estimates that he solves on the spot only 25% of user-problems. In addition, some problems (maximum of 25%) are passed to other sub-system owners for resolution.

Continuing the review of the current focused served system, Table 7.16 presents information on long-term problems for some of the sub-system owners, it shows:

- How sub-system owners record details of these long-term user-problems
- The average number of such problems that they estimate that they are dealing with at any one time
Table 7.13: Subsystem managers’ goals related to client’s goal Achieve System Availability Targets

<table>
<thead>
<tr>
<th>Role</th>
<th>Related goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unix System Manager</td>
<td>Keep Unix systems running smoothly.</td>
</tr>
<tr>
<td></td>
<td>Maximise usability of the Unix System.</td>
</tr>
<tr>
<td>Unix Computer Officer</td>
<td>To provide a good, working Unix environment.</td>
</tr>
<tr>
<td>PC System Manager</td>
<td>Provide a stable, reliable PC network.</td>
</tr>
<tr>
<td></td>
<td>Provide support for PC applications.</td>
</tr>
<tr>
<td></td>
<td>Provide a system compatible with the demands placed on it.</td>
</tr>
<tr>
<td></td>
<td>Balancing pressures for access to PC facilities against maintaining a reliable infrastructure.</td>
</tr>
<tr>
<td>PC Computer Officer</td>
<td>Ensure all laboratory PCs are “up and running”.</td>
</tr>
<tr>
<td></td>
<td>Ensure all PC software is available on laboratory machines.</td>
</tr>
<tr>
<td></td>
<td>Ensure all academic PCs are “up and running that all the appropriate software is available.</td>
</tr>
<tr>
<td>Oracle System Manager</td>
<td>No goals stated.</td>
</tr>
<tr>
<td>Small Systems Systems Manager</td>
<td>Make sure that computer laboratories are usable by academic staff (highest priority) and students (next priority).</td>
</tr>
<tr>
<td>Helpdesk Systems Manager</td>
<td>Not applicable.</td>
</tr>
</tbody>
</table>

- The criteria that they use to determine the order in which they tackle these problems
- How the users are kept informed of the status of their problems

The table shows first that most sub-system owners estimate that they are dealing with ten to twenty long-term user-problems at any one time. Second, it also shows that they use a variety of methods for maintaining an awareness of these problems. These methods range from the use of memory, through the use of formatted (“Helpdesk forms”) and unformatted (“Offline to-do list”) records written with paper and pen, to the use of computer-based system such as “on-line spread sheets” and “e-mail lists”. Third, it shows that, generally, all sub-system owners inform users of the status of their problems at significant moments in the history of their resolution, for example when a required piece of software arrives or when a problem is completely resolved. Fourth, and finally, it shows that unresolvable problems tend to be placed on the “back-burner”; that is, no work is done on them unless perhaps an opportunistic discovery of new information suggests the possibility of a resolution. Some of these back-burner problems eventually disappear. For example, some disappear when the problem owner leaves the University.
Table 7.14: Subsystem managers’ goals related to client’s goal deploy Best Kit

<table>
<thead>
<tr>
<th>Role</th>
<th>Related goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unix System Manager</td>
<td>Forecast and meet Unix related needs.</td>
</tr>
<tr>
<td>Unix Computer Officer</td>
<td>No goals stated.</td>
</tr>
<tr>
<td>PC System Manager</td>
<td>Looking at long-term picture (one year ahead) and acting accordingly.</td>
</tr>
<tr>
<td>PC Computer Officer</td>
<td>Investigate new PC hardware and software technology with a view to bringing them in-house.</td>
</tr>
<tr>
<td>Oracle System Manager</td>
<td>Introduce NT to the faculty.</td>
</tr>
<tr>
<td></td>
<td>Keep Oracle up-to-date with new releases.</td>
</tr>
<tr>
<td></td>
<td>Install NT on Administration staff machines.</td>
</tr>
<tr>
<td>Small Systems Systems Manager</td>
<td>No goals stated.</td>
</tr>
<tr>
<td>Helpdesk Systems Manager</td>
<td>No goals stated.</td>
</tr>
</tbody>
</table>

Still continuing the review of the current focused served system, the requirements engineer reviewed the subsystem owners’ problems with their management of long-term user-problems. Table 7.17 presents for some of the sub-system owners’ their view of the problems they have with their own methods of maintaining an awareness of long-term user-problems. In addition it records sub-system owners’ suggestions for improving the management of long-term user-problems. A list of (paraphrased) problem descriptions ordered by decreasing importance is shown below:

- There are a number of local systems for recording and tracking user-problems rather than one integrated global system
- The local systems are imperfect: some problems are lost or forgotten
- It is hard to prioritise long-term problems
- It is difficult to track the status of long-term user-problems
- There is no provision for producing statistics on long-term user-problem management
- There is no active and little passive guidance from the local systems
- Some of the local systems perform poorly

The requirements engineer reviewed the improvements to the management of long-term user-problems that were suggested by the sub-system owners. It seemed that each improvement constituted an aspect of one of three more general improvements. These three general improvements are listed below (the numbers in brackets refer to the improvements enumerated in Table 7.17).

154
Table 7.15: Problem resolution

<table>
<thead>
<tr>
<th>Role name</th>
<th>Solved immediately (%)</th>
<th>Passed on (%)</th>
<th>Added to list of long-term problems (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unix System Manager</td>
<td>25</td>
<td>25</td>
<td>50</td>
</tr>
<tr>
<td>Unix Computer Officer</td>
<td>80</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>PC Systems Manager</td>
<td>90</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>PC Computer Officer</td>
<td>90</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>Oracle Systems Manager</td>
<td>85</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>Small Systems</td>
<td>75</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>Helpdesk Systems Manager</td>
<td>90</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

- The sub-system owners should have more interrupt-free time for working on long-term user-problems and for strategic work (improvements: 3,4,6,7,10,12,16,17, and 25)
- The sub-system owners should be able to control the way that they receive user-problems (improvements: 2,5,14,21,22, and 23)
- The sub-system owners would like there to be a centralised computer-based problem management system (improvements: 1,13,14,18) with the following functionality:
  - automated, overridable task prioritiser/scheduler (improvements: 1,19,24)
  - provision of re-usable solutions to known problems (improvement: 26)
  - support for information access, searches and ad hoc querying (improvement: 8)
  - support for statistics generation (improvement: 15)
  - automatic reminder of flagged tasks (improvement: 20)
  - support for merging tasks (improvement: 20)

Step-three: Validate each subsystem owner’s view of their subsystem: This step was not carried out because it was assumed that the correct information had been elicited during the open interviews. These were conducted in the earlier step “Elicit and Model Each Subsystem Owners’ View of their Subsystem” (stage-three, step-two).

Step-four: Observe and interview served system workers at work: The method of Contextual Inquiry [HB93] was described in Chapter Three. The purpose of the observation part of the method is to obtain a clearer idea of the detailed nature of the current work and of the kind of computer-based system features that the workers deem feasible and sensible to introduce.
Table 7.16: Long-term problems

<table>
<thead>
<tr>
<th>Role name</th>
<th>How recorded</th>
<th>Average number</th>
<th>Order tackled</th>
<th>Unresolvable</th>
<th>User informed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unix SM</td>
<td>Post-it notes</td>
<td>12</td>
<td>Numbers affected</td>
<td>Do not exist</td>
<td>Significant events in problem resolution.</td>
</tr>
<tr>
<td></td>
<td>Off-line log book</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unix CO</td>
<td>E-mail list</td>
<td>No data</td>
<td>Numbers affected</td>
<td>Transferred</td>
<td>Significant events in problem resolution.</td>
</tr>
<tr>
<td></td>
<td>Help-desk forms</td>
<td></td>
<td>Part-timers</td>
<td>Back-burner</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Meeting minutes</td>
<td></td>
<td>Face-to-face users</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Offline to-do list</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC SM</td>
<td>Offline flip pad</td>
<td>10 - 20</td>
<td>Numbers affected</td>
<td>Disappear</td>
<td>Significant events in problem resolution.</td>
</tr>
<tr>
<td></td>
<td>Online database</td>
<td></td>
<td>System stopped</td>
<td>Eventually resolved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Online files</td>
<td></td>
<td>Known quick fix</td>
<td>Back-burner</td>
<td></td>
</tr>
<tr>
<td>PC CO</td>
<td>Memory</td>
<td>No data</td>
<td>Severity per individual</td>
<td>No data</td>
<td>Significant events in problem resolution.</td>
</tr>
<tr>
<td></td>
<td>Notepad</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E-mail list</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Online spreadsheet</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle SM</td>
<td>Online spreadsheet</td>
<td>20</td>
<td>Number affected</td>
<td>Contact proprietor</td>
<td>Significant events in problem resolution.</td>
</tr>
<tr>
<td></td>
<td>Online to-do list</td>
<td></td>
<td>User’s priority</td>
<td>Back-burner</td>
<td></td>
</tr>
<tr>
<td></td>
<td>E-mail list</td>
<td></td>
<td>Irritating problems</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Offline log</td>
<td></td>
<td>High-status users</td>
<td>Short problems</td>
<td>Enduring problems</td>
</tr>
</tbody>
</table>
Table 7.17: Problems with managing long-term problems, and suggested improvements

<table>
<thead>
<tr>
<th>Role name</th>
<th>Problems with long-term problem system</th>
<th>Improvements to long-term problem system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unix SM</td>
<td>1 Possible lost problems</td>
<td>1 CBS supporting problem scheduling</td>
</tr>
<tr>
<td></td>
<td>2 Too reactive</td>
<td>2 Receive problems via Helpdesk</td>
</tr>
<tr>
<td></td>
<td>3 Sensitive to personal pressure</td>
<td>3 Reduce involvement with users</td>
</tr>
<tr>
<td></td>
<td>4 Ability to cut off from users</td>
<td></td>
</tr>
<tr>
<td>Unix CO</td>
<td>4 Some long-term tasks never started</td>
<td>5 Receive tasks in controllable way</td>
</tr>
<tr>
<td></td>
<td>5 Interrupt driven</td>
<td>6 Work only on long-term tasks</td>
</tr>
<tr>
<td></td>
<td>6 Too ad hoc</td>
<td>7 Scheduled times for interrupt-free</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8 Ability to combine related tasks</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 Reduce number of problems to zero</td>
</tr>
<tr>
<td>PC SM</td>
<td>7 Some problems lost</td>
<td>10 FAQ available for users</td>
</tr>
<tr>
<td></td>
<td>8 Information forgotten/overlooked</td>
<td>11 Centralised system</td>
</tr>
<tr>
<td></td>
<td>9 Ad hoc prioritisation</td>
<td>12 More time for strategic work</td>
</tr>
<tr>
<td></td>
<td>10 No statistics</td>
<td></td>
</tr>
<tr>
<td>PC CO</td>
<td>11 Tracking status of problems</td>
<td>13 Web-based, centralised problem-management system</td>
</tr>
<tr>
<td></td>
<td>12 Tracking problems managed by</td>
<td>14 Faults logged to centralised DB; solutions e-mailed</td>
</tr>
<tr>
<td></td>
<td>multiple people</td>
<td>15 Support for statistics generation</td>
</tr>
<tr>
<td></td>
<td>13 Cannot produce statistics</td>
<td>16 Bookable times for user visits</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17 Each CO’s capabilities published</td>
</tr>
<tr>
<td>Oracle SM</td>
<td>14 A variety of unintegrated systems</td>
<td>18 Global, homogeneous system supporting information access, searches, ad hoc querying</td>
</tr>
<tr>
<td></td>
<td>15 Cannot “see” others’ work on related systems</td>
<td>19 Manually schedulable tasks</td>
</tr>
<tr>
<td></td>
<td>16 NT: spreadsheet slow-loading and slow to enter new problem</td>
<td>20 Automated flagged task reminder</td>
</tr>
<tr>
<td></td>
<td>17 Unix: No active guidance and insufficient information stored</td>
<td>21 Tasks received via e-mail</td>
</tr>
<tr>
<td></td>
<td>18 PC: unformatted pages and pages easily lost</td>
<td>22 Tasks directed to Helpdesk first</td>
</tr>
<tr>
<td>Small systems</td>
<td>19 Some problems forgotten</td>
<td>23 Receive problems in controllable way</td>
</tr>
<tr>
<td></td>
<td>20 Hard to organise/prioritise problems</td>
<td>24 Automated, overridable task scheduler/prioritiser</td>
</tr>
<tr>
<td></td>
<td>21 Hard to say “no” to users</td>
<td>25 Means of notifying users of availability</td>
</tr>
<tr>
<td>Helpdesk</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>


This understanding of current work is necessary because it is from this that
new ways of working will have to be introduced. The understanding of
what computer-based system elements workers think may and may not be
included is important, since it is derived from the actual workers’ experiences
of performing the work.

The Contextual Inquiry method's initial phase of observation and question-
ing has been integrated into the synthesised approach in order to facili-
tate the investigation of the focused served system. And in this case study,
six hours was spent closely observing and recording the activities of the three
COTS groups that constituted the COTS problem management system at
that time (two hours per group): the Helpdesk, User Support, and System
Support. In each case, the following process was enacted:

- Describe the session’s context
- Observe and question the selected role-holders
- Record problems identified in the session
- Summarise work performed in the session
- Analyse the session’s data
- Review the problems identified in the session
- Record requirements for new work and new computer-based systems

It was expected that analysis of the observations and also of the answers
to questions posed by the requirements engineer would lead to insights into
the kind of computer-based systems (and thus to their requirements) that
would and would not support these staff in their work. The transcripts
of the observation sessions, the questions and answers, the analyses of the
transcripts, and the derived requirements are recorded in Appendix N. Part
of the transcript, the analysis, and the derived requirements for the User
Support role are presented below to illustrate this part of the approach.

On Tuesday the 21st of October 1997 two hours, from 1400h to 1600h,
was spent observing the activity of Stephen Mullen, a member of the User
Support Group. Stephen shared an office with three other members of the
group. The office was located a few yards from the Helpdesk area.

During the two-hour period of observation, Stephen Mullen tackled prob-
lems from two sources: his long-term problem list and problem interrupts.
When he was able to, Stephen selected a problem to work on from his long-
term problem list, which he maintained as a list of e-mails (one e-mail per
problem, generally). Each time that he was able to, he would select the
next long-term problem to work on in a dynamic way according to two main
criteria: urgent problems were tackled first, but a problem was only chosen
if he deemed that there was a sufficiently large block of time available to
him to resolve it in.

Stephen was frequently interrupted in his work. There were four main
types of interrupt: Helpdesk personnel bringing students and their problems;
User Support colleagues bringing problems; “illegal” direct approaches by
students or staff; and external sources, for example a Maintenance Company.
Stephen used the following strategy for dealing with interrupts. He would
SM selects from his e-mail list of problems the most urgent background problem to work on next: in this case it is restoring someone’s e-mail files.

SM goes to restore room in 2P46.

SM suspects that the wrong tape is loaded.

Alwyn Barry interrupts SM with a problem; he needs to restore a student’s directory. AB leaves details with SM. AB indicates that it is not an urgent problem; it can be resolved “whenever”. SM puts down on his desk the piece of paper containing the details of this problem.

SM returns to the original restore-problem; it is difficult to resolve because the backup tapes are not in an optimal order.

SM still working on the (restore) problem.

SM seeks advice from Phil Naylor a colleague in Systems Support.

PN confirms that the right machine is being used. SM thinks that he is still not using the right tape.

SM tries the tape in the machine downstairs in 2D46 - it is the last chance to find the right tape.

SM still working on the restore problem.

SM answers a call from a maintenance company.

Karen from the Helpdesk brings in a student with a query, but SM is still on the ’phone. Karen asks the caller to return. SM checks the make and model of a machine for the caller.

‘Phone call ends.

Error message: cannot read tape.

Julia Dawson (JD) brings in a user-problem with photographs. SM says, “Leave it with me”. SM puts this problem to one side; he tells me that he will do it next because it won’t take long.

HK asks SM for PN’s extension. HK asks SM to read a word.

SM returns to the ”restore” problem. SM goes to PN, in Systems Support, for advice, but PN is not in.

A student cannot enter his password to unfreeze the screen on a Resolv machine. Student and SM confer with AP on this problem. AP solves this problem.

SM tackles JD’s photograph problem, but finds that he needs a book that is at home, so he suspends this problem.

Back to the ”restore” problem. SM e-mails the problem owner that this restore cannot be performed. The restore problem is thus terminated.

Transcript continues on the next page.
SM selects the disc space problem from his e-mail list of problems to work on. There are only two or three other problems on this list; one will be time-consuming so SM will leave it ’til tomorrow; the others are not very urgent.

SM solves the disc space problem. SM writes to this problem’s owner.

SM tackles the next problem and solves it immediately. SM writes to this problem’s owner.

Student requests user id and password. SM attends to this request straight away. SM gives the student printout details. This problem is resolved and terminated.

first ascertain the nature of the problem. If he judged that he could solve it quickly, then he would tackle it straight away; otherwise he would write the details of the problem on a scrap of paper and tackle it later. Effectively these latter problems were being added to his long-term problem list. The requirements engineer noted that scribbling problem descriptions on scraps of paper was perhaps not the best way of recording them: a scrap of paper might easily be mislaid.

The two tables Table 7.18 and Table 7.19, summarise the key features of the two-hour period of observation. They summarise the problems brought to Stephen and show the outcome for each problem. It is worth noting that he tackled ten problems in the two hours. This compares with the thirty tackled by the Helpdesk over a similar period. Since the Helpdesk filter out many of the easier-to-solve problems, one would expect that Stephen would tackle fewer problems but that they would be harder problems. Seven of the ten problems tackled by Stephen were interrupt problems, while three were from his long-term problem list. Stephen resolved seven of the ten problems brought to him; two were added to his long-term problem list; and one student was told to return again later.

With regard to requirements for a computer-based system, Stephen did not think that a computer-based problem management system would be useful to him. He prefers paper-based descriptions of problems. He doubted whether problem information could be collected electronically. However, the requirements engineer felt that it would be useful for Stephen to have a way of recording problem details using a computer-based problem management system. Problem descriptions would be much less likely to be lost if recorded in a computer-based problem management system than if written on scraps of paper. The requirements engineer noted that his long-term problem list (e-mails) was already electronically-based. The following requirements for a computer-based problem management system were inferred:

- The computer-based problem management system should provide support for displaying a list of problems, each of which should be associated with a priority

- The computer-based problem management system should provide support for quickly and easily entering the description of a problem
### Table 7.18: Source of problems encountered in the observation period and their resolution

<table>
<thead>
<tr>
<th>Problem identifier</th>
<th>From problem list</th>
<th>Staff interrupts</th>
<th>Student interrupts</th>
<th>Added to problem list</th>
<th>Resolved</th>
<th>Told to return</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>quota</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>file</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>disc</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>query</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>photo</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>screen</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>disc</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>next</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>user id</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 7.19: Summary of problems encountered in the observation period

<table>
<thead>
<tr>
<th>Information type</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problems</td>
<td>10</td>
</tr>
<tr>
<td>Problems selected from long-term problem list</td>
<td>3</td>
</tr>
<tr>
<td>Problems resolved by User Support</td>
<td>7</td>
</tr>
<tr>
<td>Problems passed to Systems Support</td>
<td>0</td>
</tr>
<tr>
<td>Problems passed to Helpdesk</td>
<td>0</td>
</tr>
<tr>
<td>Problems unresolved</td>
<td>0</td>
</tr>
<tr>
<td>Problems to be re-reported</td>
<td>1</td>
</tr>
<tr>
<td>Problems added to problem list</td>
<td>2</td>
</tr>
</tbody>
</table>
CHAPTER 7. CASE STUDY

Application of the adapted form of Contextual Inquiry in the synthesised goal-oriented approach produced the following requirements for a computer-based problem management system. It should support different kinds of problem solver in appropriate ways. In other words, the requirements for a computer-based problem management system and typical patterns of interaction with a computer-based problem management system are likely to be different for different kinds of problem solver. For Helpdesk staff, the Contextual Inquiry-like part of the approach indicates that a computer-based problem management system should enable them to record and categorise each instance of a service request quickly and easily. In particular Helpdesk staff should not be compelled to enter descriptions of problems. On the other hand, staff dealing with more difficult problems, often channeled to them by Helpdesk staff, have different requirements of a computer-based problem management system. First, it should help them to record the details of a problem, categorise it, and assign it a priority quickly and easily. Second, it should automatically raise the priority of a problem periodically. Third, the computer-based problem management system should display on demand a problem solvers’ list of assigned problems (including their priority). Fourth, and finally, a computer-based problem management system should facilitate the capture and reuse of the details of how a problem was resolved.

Step-five: Create process models of the served system: During this step the requirements engineer creates models of the processes that comprise the current work practices of the focused served system. The rationale for creating such models is as follows. First, process models may be analysed by the requirements engineer; they may be improved, that is to say made more efficient and or effective; they may be modified to meet new or altered goals; they may be used as the source for deriving requirements for computer-based systems. Second, process models represent working practices in a form which may be readily validated by the workers And third, process models may be used as an effective medium of communication about work practices among stakeholders.

The client and requirements engineer had agreed that the COTS user-problem management system would constitute the focused served system. The current processes of this focused served system were determined by the requirements engineer and modelled using Ould’s Role Activity Diagram notation [Oul95].

Student users experiencing problems with one of the computer systems supported by COTS—Unix, PC, Mac, etc.—nearly always come in person to the Helpdesk in 3P12 and report their problem to one of the Helpdesk personnel. On rare occasions they might phone in a problem to the Helpdesk or email a problem to the Helpdesk manager. Sometimes a student would report a problem directly to a system expert, e.g. a Unix expert. However, they were not supposed to do this. Staff users sometimes report their computer-system problems in person to the Helpdesk personnel. More often they will phone, email, or visit in person the COTS member of staff who they think is most likely to help them to resolve their problem.

User-problems that cannot be resolved by Helpdesk personnel are transferred to system experts in a number of ways which are described below and illustrated in the associated diagrams. One method is illustrated in fig-
When the Helpdesk manager determines that the problem cannot be resolved by Helpdesk staff, she selects the system expert whom she feels is most likely to be able to resolve the problem and communicates the problem to him or her. When the system expert resolves the problem, he or she communicates the details of the resolution back to the Helpdesk manager, who passes them on in turn to the original problem-owner. Or the system expert may pass on the details of the resolution directly to the original problem owner. In one variation of this process, the Helpdesk manager accompanies the problem-owner on a visit to the system expert, where the problem is again recounted. In such cases the Helpdesk manager usually remains present while the expert tries to resolve the problem. In this way the spread of knowledge is facilitated. In another method (see figure 7.8) the Helpdesk manager communicates the name and location of the selected system expert to the problem-owner, and invites him or her to contact the system expert on their own. For the last two processes, if the appropriate system expert is not available, the problem owner is advised to keep returning until they do contact the recommended system expert.

In general, individual system experts have their own methods for managing long-term user-problems. However, all the methods are similar, so it
is worth examining in detail how a typical expert manages such problems. Figure 7.9 shows how a particular expert, after deciding that a user-problem could not be resolved on the spot, adds the problem to his or her set of long-term user-problems. In this case the user-problem is first assigned a priority by the expert. High priority user-problems are written up on a post-it note and stuck to the expert’s workstation screen. Low priority user-problems are treated similarly, but stuck to the border of the workstation screen. Figure 7.10 shows how this system expert schedules long-term user-problems into the day’s work. At the start of the day the set of such problems is reviewed and the day’s schedule is constructed. Then one at a time each user-problem is resolved and the solution is communicated to its owner. This subprocess continues until there are no more long-term user-problems.

These descriptions and process models are of course more simple than the reality that they describe. For example, they do not say what happens when the expert is unable to resolve a long-term user-problem. Nor do they indicate how the expert manages interrupts from other problem-owners which may occur while he or she is already engaged in resolving a long-term user-problem.
Step-six: Quantify aspects of served system (optional): In general, a requirements engineer will generate quantitative data associated with aspects of the focused served system if it seems that such data will prove useful in the subsequent enactment of later steps of the synthesised approach, in other words if it seems that it will prove useful ultimately for deriving requirements for computer-based systems. In this case study, subsystem owners were required to estimate the number of user problems that they received in unit time, and the proportion of these that became long-term user problems. However, the requirements engineer considered that it would be useful to have more precise knowledge of the flow of user problems into and around the COTS system. Precise knowledge would allow the requirements engineer to estimate the scale of the current and any future focused served system. Knowledge of the scale would be likely to impact the requirements for any future computer-based systems, e.g. the requirements for storage of user-problem details.

In order to obtain a more realistic idea of the nature of the flow of problems into and around the COTS domain, the requirements engineer carried out the following procedure. For each day of one week, the personnel on the Helpdesk and in the other COTS subsystems existing at that time recorded data about each instance of a problem. Data was collected for the flow of problems into the following COTS subsystems:

- Helpdesk
- Unix System Support (2 groups)
- PC System Support (3 groups)
The collected data included the type of a problem, e.g. “User asks for information”, “User reports a problem”, etc.; the type of the user (student, academic staff, etc.); the provenance of the problem (arrived directly, arrived via Unix Systems Support, etc.); the medium of reporting (face-to-face, telephone, e-mail, etc.); whether or not the problem was solved on the spot, and, if not, whether it was passed onto a colleague in another subsystem or added to a list of long-term problems.

A supply of pre-printed forms (see Appendix O) was distributed to the Helpdesk and to the Computer Officers responsible for each of the COTS subsystems in the list above. COTS staff were asked to complete one form for each transaction that they had with a user. At the end of the day all of the used forms were collected in.

Some of the information that was collected—Number of Problems Presented, Number Solved On-the-Spot, Number passed On, and Number Placed on Problem List—is presented in Appendix O in a series of tables, one for
each day of the observation period. The table for a typical day is displayed in Table 7.20: a number of observations may be made about its contents. First, it shows that it is the Helpdesk that receives most problems in the course of a day—one hundred and twenty-five on that day. However, second, at that time it is clear that a relatively high number of problems was also being received by other COTS subsystems. For example, one of the two Unix subsystems (Unix-1) received thirty problems in the course of the day. Twenty of these were solve on-the-spot, while two were passed on and seven added to a long-term problem list. And third it is clear that most problems received by any part of COTS are resolved on the spot, although a relatively small number of problems are either passed on to other colleagues or placed on long-term problem lists. Both these findings agree with the findings from the direct observation work and with the Computer officer’s estimates.

The collection and analysis of quantitative data in this case study has indicated the order of the total number of long-term problems that are likely to arrive each day (for example fourteen on this Tuesday). It has also indicated the proportion of this total that is likely to fall to each subsystem owner daily. In both cases the number is relatively low.

Table 7.20: Flow of problems into and around COTS: Tuesday

<table>
<thead>
<tr>
<th>Name of subsystem</th>
<th>Number of problems presented</th>
<th>Number solved on-the-spot</th>
<th>Number passed on</th>
<th>Number placed on long-term problem list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unix-1</td>
<td>30</td>
<td>20</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Unix-2</td>
<td>10</td>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>PC1</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PC-2</td>
<td>9</td>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NT</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Oracle</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Network</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Atari</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Audio visual</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Small Systems</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Helpdesk</td>
<td>125</td>
<td>115</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>198</td>
<td>169</td>
<td>15</td>
<td>14</td>
</tr>
</tbody>
</table>

Stage-four: Summarise investigations and resolve anomalies:

The enactment of stages and steps to this point yielded the following key items of information:

- The client’s *highest priority goal* is User Support. Among other aspects this involves providing “a means for users to report problems and for these problems to be investigated and if possible corrected.”

- One of the client’s *most severe problems* was Problem Logging. This
involves “Logging and tracking where a problem is currently after it has been communicated to the Helpdesk.”

- The requirements engineer and client agreed that the focused served system would be the management of user-problems and, in particular, the management of long-term user-problems

- Subsystem owners’ problems with the current management of long-term user-problems included the following:
  - There are a number of local systems rather than one integrated global system
  - The local systems are imperfect: some problems are lost or forgotten
  - It is hard to prioritise long-term problems
  - It is difficult to track the status of long-term user-problems
  - There is no provision for producing statistics on long-term user-problem management
  - There is no active and little passive guidance from the local systems
  - Some of the local systems perform poorly

- Subsystem owners’ suggested improvements to the current management of long-term user-problems included the following:
  - The sub-system owners should have more interrupt-free time for working on long-term user-problems and for strategic work
  - The sub-system owners should be able to control the way that they receive user-problems
  - The sub-system owners would like there to be a centralised computer-based problem management system with the following functionality:
    * automated, overridable task prioritiser/scheduler
    * provision of re-usable solutions to known problems
    * support for information access, searches and ad hoc querying
    * support for statistics generation
    * automatic reminder of flagged tasks
    * support for merging tasks

- For Helpdesk staff, the Contextual Inquiry-like part of the approach indicates that requirements for a computer-based problem management system should include enabling them to record and categorise each instance of a service request quickly and easily. In particular, Helpdesk staff should not be compelled to enter descriptions of problems. On the other hand, staff dealing with more difficult problems, often channelled to them by Helpdesk staff, have different requirements of a computer-based problem management system:
– It should help them to record the details of a problem, categorise it, and assign it a priority quickly and easily
– It should automatically raise the priority of a problem periodically
– It should display on demand a problem solvers’ list of assigned problems (including their priority)
– It should facilitate the capture and reuse of the details of how a problem was resolved

With regard to anomalies, the requirements engineer informed the client that the stage “Investigate and Model external Views of the Served System” had yielded the following:

- For one of the client’s less important goals, “Contribute to the University IT strategy”, no matching goal owned by any of the subsystem owners had been found
- The Dean had identified three goals for which the client had no matching goals

These anomalies were not followed up in the subsequent portion of the case study, because the client did not consider that they were sufficiently important.

7.3.2 Part-two: Investigate IT and benchmarks, and select stakeholders

Application of part-one of the approach was concerned to conceptualise the current served system, and, in particular, to identify and conceptualise the current focused served system. Part-three of the approach is concerned with deriving requirements for new served and serving systems. In part-two, the main concern is to select the stakeholders who will be involved in enacting part-three. Optionally it may also be concerned to investigate both relevant IT and relevant benchmarks. Part-two of the synthesised approach comprises three stages:

- Investigate and publish information on relevant IT (optional)
- Benchmarking (optional)
- Assemble stakeholder group

The enactment of each stage is now described in detail.

Stage-one: Investigate and publish information on relevant IT (optional):

It was not considered necessary to perform this stage, as all the stakeholders were knowledgeable about relevant, current IT.
Stage-two: Benchmarking (optional)

The requirements engineer investigated the University’s central user-problem management system. This was considered to be an existing solution with similar goals to the COTS’ user-problem management system. It was expected that aspects of this solution might be helpful to the stakeholders when they came to consider a new served and serving (user-problem management) system. In the event, the requirements engineer discovered that an old system was in the process of being replaced by a new system. The old system is described, as are the problems that led to the specification of requirements and goals for a new system. The new system is described after that, and this is followed by a list of requirements to be considered by the stakeholders in the case study.

The architecture of the old system is shown in figure 7.11. The old system operates in the following way. Users convey problems to either the University Helpdesk (UHD) or a University Computer Officer (UCO) either face-to-face, or by telephone, e-mail or letter. University Computer Officers always pass on the the University Helpdesk problems that are reported directly to them. University Helpdesk personnel contact the problem owner and draw up a statement of requirements; this specifies what the user requires. When this has been produced, the University Helpdesk staff enter it onto the old user-problem management system. After this, either University Helpdesk staff solve the problem on-the-spot and close the task, or they allocate the task to a Systems Team Leader or Systems Team Member. A Systems Team Leader who is allocated tasks allocates them, in turn, to Systems Team Members. The Systems Team Member is responsible for managing the task, liaising with the user as necessary. As he or she works on a task, a System Team Member records significant events in the task’s history in a free-form format on the system. When they have completed the task, they close it. The University Helpdesk manager monitors all tasks, and may send messages (via a task’s progress record) to a System Team Member if the task is not being progressed sufficiently quickly.

The following problems with the old system prompted the specification of requirements and goals for a new system.

- **Ownership of tasks:** It is not always clear who a task should be assigned to
- **Lack of automatic triggers:** The old system lacks automatic event triggers: for example, when the University Helpdesk has assigned a System Team Member a task, it does not automatically inform them
- **Impossible to support ad hoc querying:** The problem-progress-descriptors have a free-form format, on top of which it is impossible to build an ad hoc query service
- **Paucity of management information:** The system does not provide enough problem-tracking information for management
- **No support for the escalation of a task’s priority:** The old system cannot automatically raise the priority of a task after some prespecified condition has occurred
Because of the problems with the old system, a new system was proposed; it was based upon the goals of the old system, but, in addition, it also had the following goals:

- The new system will support a knowledge base of standard problems and their solutions, particularly for the benefit of novice users
- The new system will provide more statistics: for example, it will output the number of open-tasks per faculty, and the average time that it takes to resolve a user-problem
- The new system will automatically raise the priority of a user-problem when some prespecified condition has been satisfied

The University acquired a proprietary product as the foundation of the new user-problem management system. The new system had a similar architecture to the old one, and it operated in a similar way. The additional benefits of the new system over the old include the following:

- The new system keeps track of the history of all transactions
- The new system has Windows interface
- User-defined information may be added to the interface of the new system, and also to the underlying database
- The user interface may be configured globally and per user
- In the new system, data is held in a structured format, so it is possible to generate ad hoc queries on this data

Figure 7.11: The socio-technical architecture of the old University user-problem management system
Only the “super-user” may delete data stored in the new system.

Accompanying the benefits, however, there were a number of problems. These are listed below:

- There are no procedures to guide each type stakeholder on how to use the new system
- The System team staff feel that transaction take longer, but the University Helpdesk staff are happy with the performance of the new system
- The new system often “hangs”

From this benchmarking work, the requirements engineer exported the following goals to be considered in the case study:

- **Performance requirement**: Specific performance goals should be specified for any new COTS user-problem management system, including goals for transaction times
- **Reliability requirement**: Specific reliability goals should be specified for any new COTS user-problem management system
- **Operational procedures**: A specific goal should be stated relating to the production of procedures that explain how each type of stakeholder should interact with any new system

Since the valuable benchmarking results were obtained with just two one-hour interviews, this activity was considered to have been worthwhile.

**Stage-three: Assemble stakeholder group:**

During this stage, the requirements engineer assembled a group of stakeholders in order to enact part-three of the new approach. The role of this group was to consider goals for the new served and serving systems, to help to identify and resolve any conflict between these goals, and to suggest ways of satisfying the goals. The requirements engineer tried to ensure that the group represented the main categories of stakeholder involved in this case study. The decisions are shown below and then discussed.

- Stakeholders with a financial interest in the proposed computer-based system
  - Client
- Stakeholders who have an interest in the use of the proposed computer-based system. These may be of three kinds:
  - Predicted frequent users of the proposed computer-based system
    * Helpdesk staff
    * User Support staff
    * System Support staff
– Predicted occasional users of the proposed computer-based system
  * Client
  * Academic and administrative staff
  * Postgraduate students
  * Undergraduate students
– People who it is predicted might be otherwise affected by the proposed computer-based system
  * No stakeholders were nominated; with hindsight, perhaps the dean should have been

• Stakeholders responsible for the design of the proposed computer-based system
  – Requirements engineer
  – All other stakeholders

• Stakeholders responsible for the development of the proposed computer-based system
  – No stakeholders were nominated because the exercise was primarily concerned with just the requirements phase

• Stakeholders responsible for the deployment, administration, and maintenance of the proposed computer-based system
  – No stakeholders were nominated because the exercise was primarily concerned with just the requirements phase

In order to determine appropriate stakeholders and groups of stakeholders in each category, the requirements engineer reflected upon the problem context. At that time the management of user-problems was handled within COTS by Helpdesk personnel, by User Support personnel, and by Systems Support personnel. It seemed appropriate therefore that one person from each of these groups should be selected as a representative stakeholder for that category. In addition, it seemed appropriate that the client should also be invited to be a stakeholder, both because he had a financial interest in the proposed system, and because he was likely to be an occasional user of it. There are two main categories of COTS service users in the faculty who were likely to use the proposed computer-based system occasionally: staff and students. So the stakeholder group had to include people from each of these categories. Within the student group there are undergraduate and postgraduate groups, and within the former group first-, second-, and third-year as well as placement students. The requirements engineer chose people on the basis of their response to the User Questionnaire (see Appendix I). The more interesting their responses, the more likely they were to be selected. The requirements engineer tried to ensure that the group’s size did not exceed six (plus or minus two) people: a group within this size range is more likely to be productive than a larger or smaller group. To help to achieve this, the requirements engineer decided to select one member of
staff, one undergraduate student and one postgraduate student. The final stakeholders selected were as follows:

- Requirements Engineer
- Client
- Helpdesk staff member
- User Support staff member
- Systems Support staff member
- Academic and administrative staff member
- Postgraduate students representative
- Undergraduate students representative

### 7.3.3 Part-three: Derive requirements for new served and serving systems

During part-three, the goals for new served and serving systems identified by the client and other stakeholders are used to derive requirements for the serving system computer-based systems and for the served system. Part-three of the synthesised approach comprises five stages:

- Model stakeholders’ goals
- Identify new served and serving systems
- Identify goal conflict and outline new served and serving systems
- Resolve goal conflict and refine served and serving systems
- Document requirements for new served and serving systems

The enactment of each stage is described in detail below.

#### Stage-one: Model stakeholders’ goals:

This stage comprises four steps:

- Elicit and validate the client’s view of problems, goals, and constraints for the new served and serving systems
- Review other stakeholders’ views of problems, goals and constraints
- Transform all problems into goals
- Create goal models

The enactment of each step is now described in detail.
Step-one: Elicit and validate the client’s view of problems, goals, and constraints for the new served and serving systems: The requirements engineer interviewed the client and asked him questions about his view of problems, goals, and constraints associated with the focused served and serving systems, that is to say with managing user-problems. The client’s complete answers to these questions are presented below in three tables: a Client’s Problems Table (Table 7.21), a Client’s Goals Table (Table 7.22), and a Client’s Constraints Table (Table 7.23). The tags associated with problems, goals, and constraints are made up as follows. “C1” refers to client number one (in this case study there was only one client); “P1”, “G1”, and “C1” refer to problem number one, goal number one, and constraint number one respectively; and so on.

A report written by the requirements engineer summarising the client’s view of problems, goals, and constraints associated with managing user-problems (see Appendix P) was sent to the client for validation. The client reviewed the report and accepted it.

Step-two: Review other stakeholders’ views of problems, goals and constraints: In this case study there are two other important groups of stakeholders: the COTS personnel, who participate in the management of user-problems, and the users themselves—academic staff, administrative staff, and students. The views of the COTS staff stakeholders, both on problems with the existing user-problem management system and goals for a future user-problem management system, had already been elicited by the approach. This was done in part-one during the step “Elicit and model each subsystem owner’s view of their subsystem” (part-one, stage-three, step-two). The data for all members of COTS staff and for the users is recorded in Appendix Q. The data for two members of COTS staff and for one user of the service are tabulated in Appendix R in Stakeholder Problems Table, Stakeholder Goals Tables, and Stakeholder Constraints Tables. For illustration, the problems table and goals table for Stakeholder three are reproduced in Tables 7.24 and 7.25.

The requirements engineer derived COTS’ stakeholders view of problems with the user-problem management system from their answer to the question “What happens to problems that do not seem possible to resolve?” (Appendix E, section E.3 (see page 251), question 6 (d)) and the question “Do you think there are any problems with the system that you use to manage your subsystem’s long-term problems?” (Appendix E, section E.3, (see page 251), question 6 (f)). The answers are recorded in Appendix M (page 315). Similarly, the requirements engineer has derived COTS stakeholders’ view of goals for a future user-problem management system from their answer to the question “In an ideal world, how would you like to improve your problem management system?” (Appendix E, section E.3 (See page 251), question 7) and the question “Is there anything else that you think I should know about problem management?” (Appendix E, section E.3, (see page 251), question 8). The answers are recorded in Appendix M. The requirements engineer had already elicited the views of COTS staff on constraints associated with any new user-problem management system (new served and serving systems). This was done in part-one when the requirements engineer directly observed COTS staff at work managing user-
### Table 7.21: Case study Client’s Problems Table

<table>
<thead>
<tr>
<th>Problem identifier</th>
<th>Short problem description</th>
<th>Problem description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1P1</td>
<td>Some problems are lost</td>
<td>There should be zero “lost” user-problems. User-problems should not be allow to just fade away and eventually be forgotten just because there are no current solutions.</td>
</tr>
<tr>
<td>C1P2</td>
<td>Users find it hard to track problems.</td>
<td>Currently most user-problems are communicated to COTS by word of mouth. They are passed around COTS from one member of the team to another in the same way. Consequently, some time after reporting a problem, it is often difficult for a user to find out the current status of that problem. The chain of COTS staff who have been involved with that problem has to be uncovered in order to find the person currently dealing with the problem and thus ascertain its status.</td>
</tr>
<tr>
<td>C1P3</td>
<td>Problem frequency data is not maintained.</td>
<td>It is not possible currently to determine the frequency of occurrence of particular types of problems. And in general it’s not possible to look for any kinds of patterns in the occurrence of problems.</td>
</tr>
<tr>
<td>C1P4</td>
<td>Hard to access solutions to infrequently occurring but hard-to-fix problems.</td>
<td>It is often difficult to retrieve the details of solutions to complex, difficult, infrequently occurring user-problems each time that they are needed. Searching for the solution usually entails a COTS member of staff trying to remember the date when the problem last occurred and then searching through paper-based files in which the details of the solution might have been recorded.</td>
</tr>
<tr>
<td>C1P5</td>
<td>Prioritising problems is problematic.</td>
<td>Computer Officers are often unsure as to how to prioritise a particular user-problem. They each have certain personal criteria for making such decisions, e.g. give top priority to the user-problem that affects most people. And to a certain extent these sets of criteria overlap. However, sometimes they allow their usual criteria to be overridden by other criteria.</td>
</tr>
<tr>
<td>C1P6</td>
<td>Users receive insufficient problem status information.</td>
<td>Currently, COTS are not good at feeding back the current status of user-problems to the problem-owners.</td>
</tr>
<tr>
<td>C1P7</td>
<td>Managing problems is difficult as they have low visibility.</td>
<td>It is difficult to manage staff who are attempting to resolve user-problems when there is usually no explicit record of each user-problem and its current status to refer to. For example it is currently difficult for the COTS manager to find out which problems are currently being worked on, which problems are blocked and why, which problems are being overlooked, and so on.</td>
</tr>
<tr>
<td>C1P8</td>
<td>Problems may remain unsolved indefinitely.</td>
<td>A user-problem with a low-priority may remain unsolved indefinitely.</td>
</tr>
</tbody>
</table>
Table 7.22: Case study Client’s Goals Table

<table>
<thead>
<tr>
<th>Goal identifier description</th>
<th>Short goal description</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1G1</td>
<td>Database of common problem-solution descriptions</td>
<td>Maintain a database of common problem-solution descriptions and provide easy and efficient access for users to this information. Such a system could be an online system.</td>
</tr>
<tr>
<td>C1G2</td>
<td>Uniform user-problem management</td>
<td>Ensure that user-problems are managed in a uniform way irrespective of their system of origin—PC, MAC, Unix, etc.</td>
</tr>
<tr>
<td>C1G3</td>
<td>One rule set</td>
<td>In the computer-based component of the user-problem management system, use only one set of rules, protocols, forms, etc. for managing user-problems.</td>
</tr>
<tr>
<td>C1G4</td>
<td>Automatic priority raising</td>
<td>The priority of each user-problem recorded in the user-problems management system should be automatically raised each time a pre-specified time-period elapses.</td>
</tr>
<tr>
<td>C1G5</td>
<td>Manager monitoring</td>
<td>It must be possible for a manager to use the user-problem management system to see at any time who is working on what, and the status of all current user-problems.</td>
</tr>
<tr>
<td>C1G6</td>
<td>Regular status reports</td>
<td>Each user should be informed of the status of his or her user-problem at sensible times during its lifecycle.</td>
</tr>
<tr>
<td>C1G7</td>
<td>Inadvertently no goal was designated as G7.</td>
<td>Inadvertently no goal was designated as G7.</td>
</tr>
<tr>
<td>C1G8</td>
<td>Log faults from anywhere</td>
<td>It should be possible for users to log faults with the COTS team from a variety of geographical locations.</td>
</tr>
<tr>
<td>C1G9</td>
<td>Match problem to expert</td>
<td>COTS should provide a mechanism whereby each user-problem obtains the level of expertise it requires for its resolution as quickly as possible.</td>
</tr>
<tr>
<td>C1G10</td>
<td>Single problem owner</td>
<td>For the life of a problem it should be “owned” by one COTS team member.</td>
</tr>
<tr>
<td>C1G11</td>
<td>Support in one room</td>
<td>The User Support Team should be housed in the same room.</td>
</tr>
<tr>
<td>C1G12</td>
<td>Multi-skilled support</td>
<td>User Support team members should become increasingly able to handle problems from a variety of computer system sources—PC, Unix, MAC, etc.</td>
</tr>
<tr>
<td>C1G13</td>
<td>Multi-knowledge</td>
<td>User Support team members should become increasingly aware of each others’ areas of expertise and skills.</td>
</tr>
<tr>
<td>C1G14</td>
<td>Very happy users</td>
<td>The level of users’ happiness with the service provided by COTS should be increased until it approaches all users being “very happy”.</td>
</tr>
<tr>
<td>C1G15</td>
<td>Maintaining problem record</td>
<td>It must be possible to add details to any record of a problem throughout its lifecycle.</td>
</tr>
</tbody>
</table>
Table 7.23: Case study Client’s Constraints Table

<table>
<thead>
<tr>
<th>Constraint identifier</th>
<th>Short constraint description</th>
<th>Constraint description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1C1</td>
<td>Timescales-1</td>
<td>The bulk of the requirements should be completed by September 30th, 1998.</td>
</tr>
<tr>
<td>C1C2</td>
<td>Timescales-2</td>
<td>A complete system should be up and running by September the 30th, 1999.</td>
</tr>
<tr>
<td>C1C3</td>
<td>Budget</td>
<td>Development costs should not exceed £6,000, i.e. the cost of one placement student for one year.</td>
</tr>
<tr>
<td>C1C4</td>
<td>Staff</td>
<td>The introduction of any new system cannot depend on any increase in the number of COTS team members, nor must it cause any decrease in the numbers of COTS team members.</td>
</tr>
<tr>
<td>C1C5</td>
<td>Development methods</td>
<td>The choice of development method is not constrained.</td>
</tr>
<tr>
<td>C1C6</td>
<td>Development languages</td>
<td>The language used should be one for which there is currently expertise within COTS since the COTS team will probably maintain the system. So the candidates are the following: Perl, C, or HTML.</td>
</tr>
<tr>
<td>C1C7</td>
<td>Multi-platform integrated architecture</td>
<td>The system should work in a uniform way on a variety of system platforms—Unix, PC, MAC, and so on—and be integrated in the sense that a global view of all user-problems can be accessed and updated from any platform.</td>
</tr>
<tr>
<td>C1C8</td>
<td>Performance</td>
<td>The users of the system must be happy with the time and effort that is required to log problems.</td>
</tr>
</tbody>
</table>
Table 7.24: Case study: Stakeholder Three’s Problems Table

<table>
<thead>
<tr>
<th>Problem identifier</th>
<th>Short problem description</th>
<th>Problem description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3P1</td>
<td>Unintegrated Problem Management System (PMS)</td>
<td>The three different systems used by stakeholder three are not integrated.</td>
</tr>
<tr>
<td>S3P2</td>
<td>Poor user-problem visibility</td>
<td>Stakeholder three cannot see what other Computer Officers are doing who are working on related tasks.</td>
</tr>
<tr>
<td>S3P3</td>
<td>Poor problem-solution recording</td>
<td>Stakeholder three cannot easily tell whether a given user-problem has ever been tackled previously.</td>
</tr>
<tr>
<td>S3P4</td>
<td>Slow to load personal PMS</td>
<td>The purpose-built spreadsheet is slow to load up.</td>
</tr>
<tr>
<td>S3P5</td>
<td>Slow paging of personal PMS</td>
<td>Stakeholder three’s purpose-built spreadsheet has to be paged through one page at a time.</td>
</tr>
<tr>
<td>S3P6</td>
<td>Insufficient information in personal PMS</td>
<td>Stakeholder three’s online To-Do-List does not enable enough information about a user-problem to be stored.</td>
</tr>
<tr>
<td>S3P7</td>
<td>Personal PMS lacks prompts</td>
<td>The To-Do-List does not enable specific categories of information to be prompted for.</td>
</tr>
<tr>
<td>S3P8</td>
<td>Unformatted information in personal PMS</td>
<td>Stakeholder three’s offline notebook is unformatted.</td>
</tr>
<tr>
<td>S3P9</td>
<td>Notebook loses pages</td>
<td>Sometimes loose pages are lost from the notebook</td>
</tr>
<tr>
<td>S3P10</td>
<td>Unresolvable problems (minor)</td>
<td>Minor unresolvable problems go to the bottom of the list of problems awaiting resolution; their resolution may be consequently delayed or put off indefinitely.</td>
</tr>
</tbody>
</table>

problems and asked them questions about their work. The results of the observations (see Appendix N) and answers to the questions helped the requirements engineer to form ideas about the constraints COTS staff would place on any future system.

The requirements engineer had also elicited the views of the users—staff and students—in part-one during the step “Consider the wider served system” (part-one, stage-three, step-one), of the approach (see page 148). A questionnaire was used to elicit this and other information. The results of the questionnaire are recorded in Appendix J. Table 6 in that Appendix records both problems that users had with user-problem management, and improvements, and new services.

Appendix Q presents the COTS staff and users’ views of problems, goals, and constraints. For some stakeholders this information is tabulated in Appendix R. But before this information can be represented in goal-hierarchy models, it is necessary to transform all the stakeholders’ problems into corresponding goals. This activity is described in the next section.
Table 7.25: Case study: Stakeholder Three’s Goals Table

<table>
<thead>
<tr>
<th>Goal identifier</th>
<th>Short goal description</th>
<th>Goal description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3G1</td>
<td>Integrated global system</td>
<td>Any user-problem management system should be integrated and global: it should support user-problems from all supported computer systems; MACs, PCs, Unix, and so on.</td>
</tr>
<tr>
<td>S3G2</td>
<td>Easy access to problem information</td>
<td>In a user-problem management system it should be easy to access problem information.</td>
</tr>
<tr>
<td>S3G3</td>
<td>Easy access to problem search service</td>
<td>In a user-problem management system it should be easy to access a problem search service.</td>
</tr>
<tr>
<td>S3G4</td>
<td>Easy access to ad hoc queries</td>
<td>In a user-problem management system it should be easy to access an ad hoc problem query service.</td>
</tr>
<tr>
<td>S3G5</td>
<td>High performance</td>
<td>A user-problem management system should be high performing.</td>
</tr>
<tr>
<td>S3G6</td>
<td>Automatic task reminder</td>
<td>A user-problem management system should automatically remind COs of flagged tasks.</td>
</tr>
<tr>
<td>S3G7</td>
<td>Easy to schedule and reschedule problems</td>
<td>Each Computer Officer should be able to schedule and reschedule user-problems.</td>
</tr>
<tr>
<td>S3G8</td>
<td>Discourage personal contact</td>
<td>Users should be encouraged to e-mail and phone in their problems rather than bringing them in personally.</td>
</tr>
<tr>
<td>S3G9</td>
<td>Academic staff via helpdesk</td>
<td>Academic staff should be encouraged to bring their user-problems to the Helpdesk in the first instance.</td>
</tr>
</tbody>
</table>

Step-three: Transform all problems into goals: All problem statements may be re-expressed as corresponding goal-statements. For example the problem “phenomenon x occurs too often” may be transformed to the corresponding goal “reduce the occurrence of phenomenon x”. In this step the problems expressed by the stakeholders are transformed to corresponding goals. In the next step these goals and the others already directly expressed are assembled into goal-hierarchy models.

Appendix S contains a number of Transform Problems To Goals Tables. Each table records the transformation of a set of stakeholder’s problems into the corresponding set of goals. Each row in a table records the following data for one stakeholder: a problem identifier, a short problem description, a goal identifier, a short goal description, and a goal description. For example, one row from the client’s table is illustrated in Table 7.26.

Step-four: Create goal models: In this step, goal-hierarchy models are drawn. For this case study, the information on the goals in the hierarchies is contained in Appendix R and Appendix S. The process of creating a Stakeholder Goal Hierarchy Diagram is illustrated below for the client stakeholder. First, the client’s problem goals are laid out. This is illustrated
Table 7.26: Transforming the client’s problem descriptions to corresponding goal descriptions

<table>
<thead>
<tr>
<th>Problem identifier</th>
<th>Short problem description</th>
<th>Goal identifier</th>
<th>Short goal description</th>
<th>Goal description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1P1</td>
<td>Some problems are lost</td>
<td>C1GP1</td>
<td>No lost user-problems</td>
<td>In any new system the number of user-problems that become lost should be zero.</td>
</tr>
</tbody>
</table>

in figure 7.12. Second, the requirements engineer tries to identify dependencies between any of these goals. In this case study it is clear, for example, that goal C1GP8, “Tackle all user-problems within reasonable time period”, supports the achievement of goal C1GP1, “No lost user-problems”. The first model is then redrawn to show all the dependencies. This is illustrated in figure 7.13. Third, the requirements engineer adds the client’s goals to the model. The result is illustrated in figure 7.14. In some cases, a client’s goals and problem-goals are identical or very similar. For example, the client’s problem goal C1GP4 is: “In any new system it should be possible to access easily the details of solutions/resolutions of rare but hard-to-fix problems which have occurred before.” While the client’s goal C1G1 is: “Maintain an electronic database of common problem-solution descriptions and provide easy and efficient access to it for users.” These are descriptions of similar goals. In such cases only one of the goals is recorded in the model, but the identifier of the other is also recorded to facilitate traceability. Fourth, and finally, the requirements engineer now reviews this model and tries to infer any “missing” goals. These are goals which fit logically into the model and are deemed to be useful by the requirements engineer but which have not been explicitly stated by any stakeholder. They could be useful in terms of helping stakeholders to understand better goals they want achieved; in terms of making explicit what stakeholders think is obvious (and therefore does not need stating); or in terms of providing low-level goals, from which requirements for computer-based systems might be derived. For example, it is clear that in order to achieve a number of the client’s goals, such as “Facilitate access to problem information” and “Provide access to problem solutions”, the details of each user-problem (and eventually its solution) should always be recorded. So in this case the goal “Record details of each user-problem and its solution” is inferred by the requirements engineer. Figure 7.15 illustrates the final goal-hierarchy for the client, where all the goals inferred by the requirements engineer have been added to the model.

In practice, the requirements engineer should also create Stakeholder Goal Hierarchy Diagrams for all the other stakeholders. The Stakeholder Goal Hierarchy Diagram for Stakeholder Three is shown as an example (see figure 7.16). The intermediate diagrams can be found in appendix T.

Step-five: Create composite goal hierarchy model: In the case study, the stakeholders were provided with a set of Stakeholder Goal Hierarchy Di-
CHAPTER 7. CASE STUDY

Figure 7.12: Client’s problem goals

Figure 7.13: Client’s problem goals reorganised
Figure 7.14: Client’s problem goals and goals
Figure 7.15: Client’s Goal Hierarchy Diagram: Client’s problem goals and goals, and requirements engineer’s inferred goals
Figure 7.16: Stakeholder Three’s Goal Hierarchy Diagram: Stakeholder Three’s problem goals and goals, and requirements engineer’s inferred goals
Table 7.27: Merging Goal Hierarchy Diagrams: Decisions

<table>
<thead>
<tr>
<th>Decisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1GP7/C1GP2 subsumes S3G2 and C1GP4/C1G1</td>
</tr>
<tr>
<td>REG11 subsumes S3GP6/S3GP8</td>
</tr>
<tr>
<td>S3GP2 subsumes C1G5</td>
</tr>
<tr>
<td>C1GP1/C1G7 subsumes S3GP9</td>
</tr>
<tr>
<td>C1G3 subsumes S3G1 and C1G2/S3GP1</td>
</tr>
<tr>
<td>C1G2 subsumes S3GP1</td>
</tr>
<tr>
<td>S3G3 is added and linked to C1GP7</td>
</tr>
<tr>
<td>S3G4 is added and linked to C1GP7</td>
</tr>
<tr>
<td>S3G6 is added and linked to C1GP8</td>
</tr>
<tr>
<td>S3G7 is added and linked to REG13</td>
</tr>
<tr>
<td>S3G9 is added and linked to C1G3</td>
</tr>
<tr>
<td>S3G8 is added and linked to S3G8</td>
</tr>
<tr>
<td>REG15 is added and linked to C1G9</td>
</tr>
<tr>
<td>REG14 is added and linked to REG15 and REG11</td>
</tr>
</tbody>
</table>

agrams rather than a Composite Goal Hierarchy Model. For completeness, a description of how to create a Composite Goal Hierarchy Model is provided. It is followed by a depiction of the model. The client’s Goal Hierarchy Diagram (see figure 7.15) is the first version of the emerging Composite Goal Hierarchy Model. Stakeholder Three’s Goal Hierarchy Model (see figure 7.16) is merged with this to create the second version of the emerging Composite Goal Hierarchy Model. The merging is accomplished in the following manner. Similar or identical goals are subsumed into the most appropriate goal, the subsumed goal identifiers being retained for reference. New “goal supports goal” relationships are identified and added to the emerging model. Conflicting goals are identified and processed as described in the previous chapter. Table 7.27 shows the decisions that were made to create version two of the the emerging Composite Goal Hierarchy Model. The model is illustrated in figure 7.17. All the goals that lie below the dotted line figure 7.17 lead to requirements for the Computer-Based Problem Management System as will be seen later.

Stage-two: Identify new served and serving systems:

The main purpose of this stage is for the requirements engineer first to orient all the stakeholders towards current served and serving systems and their problems, and towards the goals and constraints identified for any new system; and, second, to encourage them to invent new ways of attenuating and/or eliminating the problems and meeting the new and pre-existing goals.

At the beginning of this stage the requirements engineer wrote a report, Report-1, “The COTS system for managing long-term user-problems and goals for a future system” for the benefit of the stakeholders (see Appendix U). The report summarises much of the work already presented in this case study. It covers the way that user-problems were managed at that time; problems with that system; and stakeholders’ goals and constraints.
CHAPTER 7. CASE STUDY

Users to be very happy
No lost user-problems
Tackle all problems in reasonable time
Regularly raise problem priority automatically
Editable database
Record details of problems and solutions
Maximise manager's effectiveness
Maximise usability of CBPMS
Minimise time to resolve user-problems
Easy to view status of all problems
Facilitate determination of problem frequency data
Facilitate access to problem information
Easy access to ad-hoc queries
Record details of each expert
Provide access to problem solutions
Log faults from anywhere
Prioritise user-problems
Automatic task reminder
easy to reschedule user-problems
Fast access to problem details
Facilitate ease of learning CBPMS use
Match problem to expert
High performance
Easy access to problem search service

Figure 7.17: Emerging Composite Goal Hierarchy Diagram
for any future system of managing user-problems. In addition, the report invited each stakeholder to both reflect upon the current system and its problems, and to consider the goals and constraints for any new system, in order to design new ways of attenuating or eliminating the problems and of meeting the new goals.

The stakeholders were invited to a meeting (Stakeholder-Meeting-1) at which the requirements engineer distributed and presented the report (Report-1). The following text, extracted from the end of Report-1, illustrates how the stakeholders were oriented towards considering the current served and serving systems and their goals, and thinking about new ways of overcoming the problems and meeting the new goals. The stakeholders were invited to:

- Consider how user-problems are currently managed by COTS
- Consider the problems perceived with this current management from each different stakeholder’s point of view
- Consider the goals that the different stakeholders have proposed for the future management of user-problems by COTS
- Think of new ways for COTS to manage long-term user-problems which solve the problems presented and satisfy the goals presented

Additionally, each stakeholder was given a set of goal-hierarchy models representing selected stakeholders (the client, stakeholder-one, stakeholder-three, and the users).

During the meeting, the Helpdesk stakeholder indicated that she was unhappy with the client’s goal of maintaining a database of common problem-solution pairings. She felt that this might reduce the overall number of visits by users to the Helpdesk. For this stakeholder, a reduction in the number of visits to the Helpdesk signaled a diminution of its importance. And it was one of this stakeholder’s implicit goals to maintain or increase the importance of the Helpdesk. This was one example both of the way that important but unspoken goals were surfaced by the approach, and of conflict between important goals.

Shortly after the meeting the stakeholders were e-mailed: the task set them was reiterated, but they were invited to e-mail their initial suggestions to the requirements engineer prior to the next meeting.

Stage-three: Identify goal conflict and outline new served and serving systems:

The main purpose of this stage is for the requirements engineer to help the stakeholders to identify both mechanisms for satisfying goals for which there is a consensus, and mechanisms for satisfying goals where consensus is lacking.

At the end of the first meeting with the stakeholders they were invited to review Report-1, “The COTS system for managing long-term user-problems and goals for a future system” (see Appendix U). The report contains the following information:

- Details of the then current long-term user-problem management system
• Details of problems with that system
• The collection of stakeholders’ goals for a new system

In addition to reviewing the report, the stakeholders were also invited to design new long-term user-problem management system components that might attenuate or eliminate the problems and satisfy the stakeholders’ goals. Appendix V contains the response from a stakeholder that the requirements engineer received after the first meeting. More suggestions were made at the meeting (Stakeholder-Meeting-2) to which the stakeholders were invited.

A record of the meeting referred to above is included in Appendix W. The main purpose of this meeting was for the stakeholders to express the various ways that they had thought of for meeting the stakeholders’ goals. The stakeholders proposed a number of ideas for ways to satisfy the stakeholders’ goals. The requirements engineer categorised these as follows:

• Kinds of problem to be supported
• Future modes of user-COTS intercommunication
• Problem-solution database
• FAQ on the world-wide web (www)
• Multi-platform use

First, with respect to the kinds of problem to be supported, the Helpdesk and User Support stakeholders thought that only long-term user-problems should be recorded and tracked in any new CBS. The requirements engineer pointed out that the COTS manager (the client), who was not present at the meeting, would like all problems to be recorded, since comprehensive problem statistics could be used to try to obtain more resources form management. But this argument was rejected by the stakeholders present. Second, with respect to future modes of user-COTS intercommunication, the User Support stakeholders forcibly expressed the view that all users should report problems in person to the Helpdesk in the first instance. They rejected the ideas that reporting problems via electronic means or reporting directly to Computer Officers might be more efficient. However, one of the student stakeholders thought that it might be useful to be able to e-mail Computer Officers directly when trying to track problem progress. Third, with respect to a problem-solution database, User Support and student stakeholders thought that it would be a good idea to maintain a database of problem-solution pairings. Fourth, with respect to a Frequently Asked Question stage on the world-wide web, they also thought that an Frequently Asked Questions document and details of short-term but frequently occurring problems could be maintained on the web. And fifth, with respect to multi-platform use, all stakeholders present wanted any new computer-based problem management system to be available on all platforms.

A secondary purpose of the meeting was to discover how difficult they had found this task. In answer to the latter question it is clear that some stakeholders found it a difficult task. Some also thought that there were too many diagrams and that some of these, particularly goal models, were
difficult to follow. Others found that the diagrams were helpful. All found the report to be clear. This feedback from the stakeholders is discussed in the next chapter, “Critical evaluation”.

The requirements engineer reflected upon the meeting for about two weeks and then wrote a report (Report-2): “COTS’ user-problem management system: alternative designs satisfying the stakeholders’ goals” (see Appendix X). The report covered the following aspects of new served and serving systems:

- COTS organisation structure
- Classifying user-problems
- Logging user-problems
- Users communicating problems to COTS
- COTS communicating with users
- The use of the computer-based problem management system to resolve user-problems
- Assigning problem priorities and scheduling user-problems

In addition, the report outlined the services that should be provided by the proposed computer-based problem management system. The report ended by inviting the stakeholders to validate its content; to reflect upon the issues of conflict that it had identified, with a view to resolving them; and to consider some important outstanding questions.

Within the list of new served and serving system aspects, some were not contentious. For example, the report indicated that a computer-based problem management system would support the recording of long-term user-problems, and that the record of a problem would include the problem-owner’s details: name, date-and-time information, problem description, etc. The complete list of the key non-controversial components of the new user-problem management system is given below.

- COTS will be organised into System Support, User Support, and a Helpdesk
- COTS staff will use a new user-problem management system to manage user-problems
- A central feature of the new system will be a global, integrated, computer-supported user-problem database
- Descriptions of all long-term user-problems will be recorded in the database
- Descriptions of the solutions associated with long-term user-problems will also be recorded in the database
- The priority of each user-problem will be raised automatically after a predetermined period of time
• COTS staff will use the problem-solution information in the database to help to resolve user-problems

However, the report also highlighted a number of areas where different groups of stakeholders were in conflict over goals. For example, the client wanted the computer-based problem management system to record short-term user-problems as well as long-term user-problems. On the one hand the client needed a record of all user-problems in order to produce comprehensive user-problem management statistics in order to strengthen his case to faculty management for an increase in resources. On the other hand, the User Support stakeholders opposed the recording of short-term user-problems in the computer-based problem management system. They felt that it would be a time-consuming task that would lead to no immediate benefits. Another source of conflict was the manner in which user-problems should be reported to COTS. Some stakeholders felt strongly that both staff and student user-problems should always be reported in person to the Helpdesk staff in the first instance. Other stakeholders felt that a different set of rules might be appropriate for staff: for example, that staff might report their user-problems to other COTS units than the Helpdesk (to User Support, for instance); and that the mode of reporting might also include e-mail and telephone for example.

As well as providing an analysis of the context of each conflict, including a goal-conflict diagram, Report-2 also presented a range of mechanisms which the requirements engineer felt might have the potential for achieving a consensus among the stakeholders. In addition, for each mechanism, it presented the requirements engineer’s view of likely additional effects of the mechanism. And, for each effect, it presented the requirements engineer’s prediction of how each stakeholder group would evaluate the effect. For example, in Report-2, the section beginning on page 448 first reviews stakeholders’ goals in this area, then identifies the existence of conflict over how users should communicate with COTS, and finally expresses it in a goal-conflict diagram (reproduced in figure 7.18). The report continues by presenting an analysis of the context of that conflict. It notes a possible need for two communication protocols: one for staff and one for students. It identifies three targets for communication problems: Helpdesk staff, User Support staff, and System Support staff. And it identifies four channels of communication: face-to-face, telephone, e-mail, and computer-based problem management system. In the report, the requirements engineer next outlines three alternative mechanisms for users to communicate user-problems to COTS. In the first, all user-problems are always presented in person at the Helpdesk. The second relaxes this rule for staff users: they can present their user-problems to any COTS target via any communication channel. And the third relaxes the rules slightly for student-users too.

The report presents next an analysis of the impact of each of these mechanisms on each relevant stakeholder goal. For example, it suggests that the first mechanism will support the User Support goal “All users—staff and students—should report their problems in person directly to the Helpdesk in the first instance”. But it will also undermine a goal proposed by the client: “It should be possible for users to log faults with the COTS team from a variety of geographical locations”. Report-2 also presents the requirements engineer’s predictions of likely additional consequences of two
of the three communication mechanisms. The requirements engineer has attempted to assess the impact of these predicted consequences on affected stakeholder groups. For example, the requirements engineer predicts in the report that if mechanism one were chosen, some staff might violate the rules and visit selected Computer Officers directly. This might lead in turn to anger and frustration in COTS staff.

The various conflict-context analyses, conflict-resolution mechanisms, and stakeholder-impact evaluations were included in the report so that each stakeholder might reflect upon the nature of the conflicts prior to the next meeting. At the next meeting attempts would be made to eliminate the conflict.

Report-2 also includes an early presentation of the services to be provided by a computer-based problem management system. The list of services is based upon the work to date. Four classes of computer-based problem management system user have been identified:

- Users
- COTS staff
- COTS manager
- Computer-based problem management system administrator

For each class, the draft set of services to be provided are presented. For example, the services provided by the computer-based problem management system to the users will include:

- Start a computer-based problem management system session
- Send problem details to COTS
- Query problem details
- Query problem-solution database (possibly)
• Print problem-solution from problem-solution database (possibly)
• End a computer-based problem management system session

The final section of Report-2 restates the non-controversial socio-technical mechanisms and invites the stakeholders to validate them. It summarises the conflicts and implicitly invites validation and reflection. Finally it poses an outstanding question: should the users have access to the problem-solution information held in the computer-based problem management system database?

Stage-four: Resolve goal conflict and refine served and serving systems:

The main purpose of this stage is for the requirements engineer to help the stakeholders to resolve any residual goal conflicts.

Prior to Stakeholder-Meeting-3, the requirements engineer received feedback on the report Report-2 from the System Support stakeholder. This feedback is presented in Appendix Y and comprises the three types of possible feedback: validation, conflict-resolution, and outstanding issues. First, the validation feedback includes a request for clarification: “What is a Staff manger?” It also includes a suggestion for basing the COTS-to-user communication protocols on automated e-mails, rather than relying upon COTS staff. The requirements engineer reflected upon this suggestion and concluded that while the system could generate e-mails and send them to the users, it would in some cases (significant checkpoints) need to be told when to send the e-mail. It would also require that a problem’s history within COTS be updated in a timely manner, so that relevant details were sent. Second, the System Support stakeholder also sent feedback on the two conflicts that were outlined in Report-2. For the short-term user-problem conflict, the stakeholder proposed an additional conflict-resolution mechanism. The stakeholder suggested that initially only the problem type, i.e. short-term problem, would be recorded. However once a certain, predefined number of reports of the same problem had been received, then a problem description and resolution description (when available) would also be recorded. Third, for the user-COTS communication conflict, the Systems Support stakeholder indicated that he preferred communication model number two. This model allows students to report problems to the Helpdesk using any available means: face-to-face, e-mail, computer-based problem management system, and so on. The stakeholder also noted that staff users would probably tend to violate any rules and go straight to whomever they considered the most appropriate Computer Officer in COTS. Fourth, and finally, the stakeholder answered the outstanding question posed in Report-2 by indicating that in his view problem-solution information should be made available for users.

Before considering the outcome of Stakeholder-Meeting-3, it should be noted that COTS now had a new manager. The new manager had agreed to take over the role of client on this case study. This new client had a different management style from the previous manager. In particular, the new client was more likely to take decisions concerning the COTS domain without consulting the other stakeholders. The impact of this is alluded to in the following description of Stakeholder-Meeting-3.
In order to validate Report-2 at the meeting, the requirements engineer had prepared a set of questions on each of the main non-controversial areas: organisation structure, logging user problems, users communicating with COTS, COTS communicating with users, solving user-problems with the CBPMS, and prioritising user-problems. For example, regarding the validation of the COTS organisation structure, the following questions were posed: “Are you happy with the new COTS organisation structure?” and “Would you like to make any changes to the new structure? And if so, why?” The questions (see Appendix Y) were distributed to the stakeholders at the meeting. Each question was considered by the stakeholders. Their responses are contained in the minutes of the meeting (see Appendix Y on page 472). Briefly the responses were as follows:

- The new COTS organisation structure is appropriate and correctly recorded
- The computer-based problem management system will not support requests for borrowing manuals and printouts, or for buying materials
- The computer-based problem management system will hold information as outlined in Report-2 plus information on an “estimated time to solution” for each user-problem
- In the computer-based problem management system data will be structured to support user queries
- COTS staff expertise will be published in the faculty
- COTS communication with users will be as outlined in Report-2
- Querying by users of problem-solution information in computer-based problem management system will be postponed
- Problem priorities will be automatically raised periodically

The priority categories listed in Report-2 were not accepted by the client, who agreed to design new ones.

Another item on the meeting’s agenda (see Appendix Y) was “Conflict”. To address this item, the requirements engineer distributed two short papers (see Appendix Y on page 468) covering the short-term user-problem conflict and the users-communicating-with-COTS conflict respectively. Both papers describe the context of the conflicts and were intended to remind the stakeholders of the range of resolution mechanisms proposed to resolve the conflict. The paper on the short-term user-problem conflict also incorporated the suggestion made by the System Support stakeholder. As the minutes of the meeting show (see Appendix Y), it was decided that short-term user-problems would not be logged in the computer-based problem management system. In addition it was decided that both staff and student users should normally contact the Helpdesk in face-to-face mode in the first instance. However, hard-and-fast rules would not be prescribed; choice of channel and target of communication would be left to the discretion of the users—they would be expected to use their common sense when choosing.

It was evident at the meeting that the client was not going to brook any opposition to what he wanted. Thus while the Helpdesk and User Support
stakeholders agreed with his decisions, the requirements engineer had the impression that other stakeholders did not but had decided to keep quiet. Although the client had dominated the decision making at the meeting, the requirements engineer continued to apply the approach in the case study. It was felt that it was important to work with what had actually happened in the case study as well as to allow all the stages and thus the approach to be illustrated. It is worth pointing out that by this stage all conflict had been “resolved” and the non-controversial design mechanisms had either been accepted as they were described in Report-2 or refined. In the next stage the complete set of requirements for the new served and serving systems, including services to be provided by the CBPMS, are shown.

**Stage-five: Document requirements for new served and serving systems:**

During this stage the requirements for serving system computer-based systems are derived. The procedure for achieving this was specified in the previous chapter. In following this procedure, the requirements engineer first re-read the following material: the report “Report-2: COTS’ user problem management system: alternative designs satisfying the stakeholders’ goals” (Appendix X) and the minutes of Stakeholder-Meeting-3 (Appendix Y). Thus the Requirements Engineer became re-acquainted with the agreed non-controversial mechanisms for meeting stakeholders’ goals, with the resolutions of goal conflicts, and with the resolutions of outstanding issues. Then, continuing to follow the procedure, the Requirements Engineer examined the leaf-goals of the Composite Goal Hierarchy Model (see figure 7.17 on page 187). First, the leaf-goals were examined, and then goals higher up the hierarchy. At each goal, the requirements engineer asked and answered the following question: “What functionality should be provided by a computer-based system in order to satisfy this goal?” For example, after examining goal REG11, “Record details of each user-problem and solution”, and asking the question, the answer was expressed as the three requirements, R 1.1, R 1.2 and R 1.3 that are recorded in Table 7.28. While answering the question, the requirements engineer noted in the table that the computer-based problem management system was not to support “short-term user-problems”. Also at this time, the requirements engineer wondered what the user-interface of the required computer-based problem management system should look like. This was noted in the table as a question that should later be put to the stakeholders.

In addition to deriving the requirements for serving system computer-based systems, a model of the served system organisational structure was produced. This is presented in figure 7.19 as a System Map augmented with “preferred” and “allowed” access paths for different channels of communication.

### 7.4 Summary

The new synthesised goal-oriented approach to requirements engineering was successfully applied in a case study to a real-world situation—a university faculty technical support department. This chapter documents that case
Table 7.28: Stakeholders’ goals and corresponding requirements

<table>
<thead>
<tr>
<th>Goals and requirements</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goal</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>REG11 Record details of each user problem and solution.</td>
<td>R 1.1</td>
</tr>
<tr>
<td>C1GP5 Prioritise user-problems.</td>
<td>R 2</td>
</tr>
<tr>
<td>C1G4 Regularly raise problem priority automatically.</td>
<td>R 3.1</td>
</tr>
</tbody>
</table>

Table continues on the next page
<table>
<thead>
<tr>
<th>Goal number</th>
<th>Description</th>
<th>Requirement number</th>
<th>Description</th>
<th>Notes (N) and questions (Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1G15</td>
<td>Editable problem database.</td>
<td>R 4.1</td>
<td>COTS staff will be able to maintain details of a user-problem.</td>
<td>N4: COTS staff user will be a built-in user-type.</td>
</tr>
<tr>
<td>REG10</td>
<td>COTS staff maintain user-problem details.</td>
<td>R 4.2</td>
<td>COTS staff will be able to delete a user-problem.</td>
<td>Q5: which COTS staff will be able to delete user-problems?</td>
</tr>
<tr>
<td>C1G5/ C1GP6</td>
<td>Users updated sensibly.</td>
<td>R 5.1</td>
<td>The CBPMS will notify the problem-owner immediately when the details of a user-problem are first logged.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>R 5.2</td>
<td>The CBPMS will notify the problem-owner when the status of a user-problem changes.</td>
<td>N5: “status” is an attribute of user-problem. N6: values for status are presented in Report-2.</td>
</tr>
<tr>
<td>C1G8</td>
<td>Log faults from anywhere.</td>
<td>R 6</td>
<td>The CBPMS will have a client-server architecture. COTS staff and staff and student users may use the CBPMS to enter user problems.</td>
<td>N7: staff users and student users are built-in user types. N8: normally all users shall report problems to the Helpdesk in the first instance.</td>
</tr>
</tbody>
</table>
## CHAPTER 7. CASE STUDY

### Goals and requirements (continued)

<table>
<thead>
<tr>
<th>Goal number</th>
<th>Description</th>
<th>Requirement number</th>
<th>Description</th>
<th>Notes (N) and Questions (Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1G3</td>
<td>CBPMS provides uniform problem management.</td>
<td></td>
<td></td>
<td>N9: see requirements R1 to R6</td>
</tr>
<tr>
<td>C1G10</td>
<td>One problem solver.</td>
<td>R 7.1</td>
<td>One member of COTS staff will be associated with each user-problem at any one time.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>R 7.2</td>
<td>The CBPMS will maintain a list in chronological order of COTS staff who have been associated with a user-problem.</td>
<td>N10: provides an audit trail.</td>
</tr>
<tr>
<td>C1G9</td>
<td>Match problem to expert.</td>
<td></td>
<td></td>
<td>N11: See R7 Q6: who will allocate logged user-problems to individual members of COTS staff.</td>
</tr>
<tr>
<td>C1GP7/C1GP2</td>
<td>Facilitate access to problem information.</td>
<td></td>
<td></td>
<td>N12: see requirements R1 and R6. N13: The manager will be a built-in user type.</td>
</tr>
<tr>
<td>C1GP4/C1G1</td>
<td>Provide access to problem solutions.</td>
<td></td>
<td></td>
<td>N14: see requirements R1 and R6. N15: staff and student users will not have access to solutions until further notice.</td>
</tr>
</tbody>
</table>

Table continues on the next page
### Goals and requirements (continued)

<table>
<thead>
<tr>
<th>Goal number</th>
<th>Description</th>
<th>Requirement number</th>
<th>Description</th>
<th>Notes (N) and questions (Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1GP3</td>
<td>Facilitate the determination of problem frequency data.</td>
<td>R8</td>
<td>The CBPMS will provide statistical information on user problems on demand.</td>
<td>Q7: which user types will have access to this service? Q8: what statistical information will need to be provided? Q9: how should statistical information be presented?</td>
</tr>
<tr>
<td>Users goal 13 (see Appendix Q on page 377).</td>
<td>Publicise skill profiles for each member of the COTS team.</td>
<td>R9</td>
<td>The CBPMS will provide details of the expertise of COTS staff on demand.</td>
<td>N16: the CBPMS will not support a &quot;preferred times of contact&quot; attribute.</td>
</tr>
<tr>
<td>S3G4</td>
<td>Easy access to ad hoc queries.</td>
<td>R10</td>
<td>The CBPMS will provide an ad hoc query service.</td>
<td></td>
</tr>
<tr>
<td>S3G2</td>
<td>Easy access to problem search engine.</td>
<td>R11</td>
<td>The CBPMS will provide a search facility for users to query long-term user-problems on key words.</td>
<td>Q10: Is this the same requirement as R10?</td>
</tr>
<tr>
<td>REG15</td>
<td>Record details of each expert.</td>
<td>R12</td>
<td>The CBPMS will permanently store details of an expert until they are deleted.</td>
<td></td>
</tr>
</tbody>
</table>

Table continues on the next page
<table>
<thead>
<tr>
<th>Goal number</th>
<th>Description</th>
<th>Requirement number</th>
<th>Description</th>
<th>Notes (N) and questions (Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3G6</td>
<td>Automatic task reminder.</td>
<td>R13</td>
<td>The CBPMS will provide an automatic task reminder service to remind experts of their scheduled problems.</td>
<td>Q11: What will the reminder interval be? Will it be configurable for each expert?</td>
</tr>
<tr>
<td>S3G7</td>
<td>Easy to reschedule problems.</td>
<td>R14</td>
<td>The CBPMS will provide a service for scheduling and rescheduling problems.</td>
<td>Q12: Will it be available to all types of CBPMS users? Or will its availability be configurable?</td>
</tr>
<tr>
<td>S3GP5/ S3GP4</td>
<td>Fast access to problem details.</td>
<td>R15</td>
<td>The CBPMS will provide fast access to problem details.</td>
<td>Q13: How fast is fast?</td>
</tr>
</tbody>
</table>

study. In the next chapter the work undertaken in support of this thesis is critically evaluated.
Figure 7.19: System Map of the new served system
Chapter 8

Critical evaluation
8.1 Introduction

Problems with both traditional approaches to requirements engineering and with three more-recent socio-technical approaches are presented in Chapter Two and Chapter Three respectively. Problems associated with two established goal-oriented approaches are presented in Chapter Four. A synthesised goal-oriented approach to requirements engineering designed to attenuate or eliminate these problems is presented in Chapter Six. The approach is based upon a framework of principles and features for building requirements engineering methods which was established from a critical review of established approaches. The extent both to which the new approach compares to established approaches, and to which problems of the latter have been attenuated or eliminated by the new approach, is assessed below in the section “Evaluation of the approach”. The assessment of the former is made using the framework; the assessment of the latter is based upon a consideration of both the details of the approach—its activities and models—and an application of the approach in a substantial case study (see Chapter Seven). In the section “Evaluation of the contribution of the thesis”, an assessment is made of the extent to which the contribution of this thesis, presented in Chapter One, has been substantiated.

8.2 Evaluation of the approach

This section presents the two ways in which the new synthesised goal-oriented approach was evaluated. First, the framework of principles and features developed in Chapter Six was used to assess both the new approach and the established approaches that had been critically reviewed earlier. The results of the assessment were used to enable the new synthesised goal-oriented approach to be compared to the established approaches. This work is presented in the next section, “Using the framework to evaluate the new approach”. Second, a critical assessment is made of the extent to which the new synthesised goal-oriented approach addresses the key requirements engineering problems identified in earlier chapters. This work is presented in the section “Addressing key requirements engineering problems”.

Using the framework to evaluate the new approach

Critical review of selected established socio-technical, goal-oriented, and process-oriented approaches to requirements engineering led to the development of a framework of principles and features that could be used both to inform the design of new approaches, and to assess, and thus to compare, approaches. The framework was presented in Chapter Six and is reproduced below. The new synthesised goal-oriented approach and the established approaches were assessed using the principles and features framework. The results of the assessment are summarised in Table 8.1. An “H” in the table means that the principle or feature is present to significant degree, an “M”, to a moderate degree, an “L”, to a low degree, and an “N” means it is not present at all. An “?” means that it is not clear whether or not it is present and an “I” means that its presence is implied but not explicitly stated.

In the framework, the general requirements engineering method principles are as follows. A method should:
• Understand the current broad served system first (conceptualise current first)

• Use an understanding of the broad served system to select a focused served system to improve (broad before focused)

• Conceptualise the focused served system before its associated serving system (served before serving)

• Encourage the proactive, cooperative participation of stakeholders (proactive, cooperative stakeholders)

• Maximise the use of validation (maximise validation)

• Be clearly articulated in terms of a range of levels of abstraction, from relatively abstract—parts and stages, for example—to relatively detailed—steps and activities, for example (clearly articulated method)

• Produce appropriate models, each defined with a clear syntax and semantics (well defined models)

• Encourage in stakeholders an awareness of the state-of-the-art in relevant areas of IT (IT aware stakeholders)

In addition to these general principles, the framework comprises the following features derived from the established approaches that were critically reviewed in earlier chapters:

• Conceptualise goals and problems of the current broad served system (current goals and problems)

• Conceptualise the current work of the broad served system (current work)

• Conceptualise CBS support for current goals and work (current CBS support)

• Identify stakeholders’ new goals for focused served system and resolve conflicting goals (stakeholders’ new goals)

• Design new work to satisfy goals (new work)

• Design alternative CBS solutions to support new goals and work (alternative CBSs)

• Predict and assess impact of side effects of alternative CBSs (impact of side effects)

• Choose from among alternative CBS solutions and document requirements (document CBS requirements)

It is clear from the table that the new synthesised goal-oriented approach to requirements engineering scores highly with respect to the framework principles and features. Each of the established approaches scores less highly.
Table 8.1: Using the framework to compare the new and established approaches

<table>
<thead>
<tr>
<th>Principles</th>
<th>Socio-technical</th>
<th>Goal-oriented</th>
<th>Process-oriented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptualise current first</td>
<td>H</td>
<td>N</td>
<td>M</td>
</tr>
<tr>
<td>Broad before focused</td>
<td>H</td>
<td>N</td>
<td>M</td>
</tr>
<tr>
<td>Served before serving</td>
<td>H</td>
<td>N</td>
<td>H</td>
</tr>
<tr>
<td>Proactive, cooperative</td>
<td>H</td>
<td>M</td>
<td>L</td>
</tr>
<tr>
<td>Maximise validation</td>
<td>H</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>Clearly articulated method</td>
<td>H</td>
<td>L</td>
<td>M</td>
</tr>
<tr>
<td>Well defined models</td>
<td>H</td>
<td>M</td>
<td>L</td>
</tr>
<tr>
<td>IT aware stakeholders</td>
<td>H</td>
<td>L</td>
<td>N</td>
</tr>
<tr>
<td>Features</td>
<td>H</td>
<td>L</td>
<td>M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Final areas</th>
<th>Socio-technical</th>
<th>Goal-oriented</th>
<th>Process-oriented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current goals and problems</td>
<td>H</td>
<td>N</td>
<td>M</td>
</tr>
<tr>
<td>Current work</td>
<td>H</td>
<td>N</td>
<td>M</td>
</tr>
<tr>
<td>Current CBS support</td>
<td>M</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Stakeholders’ new goals</td>
<td>H</td>
<td>N</td>
<td>L</td>
</tr>
<tr>
<td>New work</td>
<td>H</td>
<td>N</td>
<td>M</td>
</tr>
<tr>
<td>Alternative CBSs</td>
<td>H</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>Impact of side effects</td>
<td>H</td>
<td>H</td>
<td>M</td>
</tr>
<tr>
<td>Document CBS requirements</td>
<td>H</td>
<td>L</td>
<td>M</td>
</tr>
</tbody>
</table>

205
Addressing key requirements engineering problems

In the early chapters of this thesis a number of problems are presented with traditional approaches to requirements engineering, with more-recent socio-technical approaches, and with two well established goal-oriented approaches. Section 2.5 (see page 23) presents a list of problems associated with traditional approaches to requirements engineering. The list is reproduced below:

- There is often a neglect of the wider organisational context [McD94, Eas88, Fly98]
- The high-level objectives of an organisation are often ignored [McD94]
- Business needs are often ignored [KS98]
- There is often a poor understanding of the application domain [CKI88, KS98]
- There is often a low involvement of the stakeholders [KS98]
- There is often poor communication among stakeholders [Mac96]
- Stakeholders often resist new systems [Fly98, KS98]
- There is often a misunderstanding users’ work [Mac96, LHG94]
- Often the wrong problem solved [Fly98]
- There are often negative consequences of new systems
- Ineffective systems are often produced [Fly98]

Problems with three more recent socio-technical approaches to requirements engineering are presented in section 3.3 (see page 45). The problems are recapitulated here:

- The descriptions of the approaches lack detail
- The descriptions of some of the approaches are incomplete
- The notations used for models used in some of the approaches lack an adequate syntax and semantics.

With regard to the established goal-oriented approaches, the main problem with the KAOS [DFvL91] approach is that its use of formal notations may mean that requirements engineering practitioners may be reluctant to use it in a real-world context. The two main problems with Loucopoulos’s approach [KLF96] are as follows. First, it may be difficult for a requirements engineering practitioner to apply because it is not described in great detail. Second, it assumes that there is a close relationship between the goals of an enterprise and the personal goals of its members. However, this may not always be a valid assumption: in some organisations, members may have goals that differ from the official enterprise goals, and from each other.

\[1\]Page 20 and Chapter 4
The extent to which the problems presented above are likely to be attenuated or eliminated by the new approach has been considered. The results of this consideration are presented below. The consideration is based upon both the approach—its activities and models (see Chapter Six)—and upon the experience of applying the approach in a case study (see Chapter Seven).

One group of problems is associated with understanding the broadest sense of the served system. First, there is often a neglect of the wider organisational context [McD94, Eas88, Fly98]. Requirements engineering methods typically elicit the requirements for a specific application and ignore the wider organisational context. In contrast to this, the new approach explicitly elicits information about the wider organisational context. It does so by eliciting information from a client about the client’s domain, and from the client’s subordinates about their part of the client’s domain (part-one, “Conceptualise the served system”). In addition, it explicitly elicits from a client’s manager the manager’s perspective on the main goals of the client’s domain (stage 1.2, “Investigate and model external views of the served system”). Stage 1.2 could be extended to model more features—structure, problems, tasks, and so on—of the immediate external domain, and extended still further to investigate and model higher levels of an organisation.

Second, part of the neglect referred to above concerns the high-level objectives of an organisation, which are often ignored [McD94]. But the new approach explicitly elicits the high-level goals of a client from the client’s perspective (stage 1.1.2 “Elicit and model client’s view of the served system”) and from the client’s manager’s perspective (stage 1.2 “Investigate and model external views of the served system”). However, the extent to which a client’s goals correspond to “the high-level objectives of an organisation” will depend upon the position of the client’s domain within the organisation. To ensure that the high-level objectives of an organisation are always elicited, the approach must be modified to support this requirement.

Third, another part of the neglect concerns business needs, which are often ignored [KS98]. Business needs exist at different levels within an organisation. At the highest level, they correspond to the high-level objectives of an organisation. At lower levels, they correspond to the goals of subsystems within the organisation. The new approach elicits a client’s goals, and thus the business needs of the organisation, at the level at which the client operates. However, to ensure that all the relevant business needs are considered, the approach must be modified to incorporate a new stage that does this.

Finally, there is often a poor understanding of the application domain, but the new approach explicitly investigates and represents a number of features of the application domain—structure, goals, tasks, problems, improvements, and so on. Engaging in this investigation should lead to a requirements engineer attaining a good understanding of an application domain.

Another group of related problems is associated with stakeholders. First, there is often a low involvement of stakeholders in requirement engineering methods [KS98]; however, in the new approach the client stakeholder and the client’s subordinates (also stakeholders) are explicitly required to be involved in a requirements engineer’s investigation of the client’s domain (stages 1.1.2, “Elicit and model client’s view of the served system”, and
In addition, in part-three of the new approach, stakeholders work closely with a requirements engineer in a series of meetings to identify goals of a future served and serving system, and also to identify alternative ways of meeting those goals.

Second, even when stakeholders are explicitly involved, there is often poor communication among them. In typical requirements engineering methods a requirements engineer elicits requirements from stakeholders on an individual basis. The stakeholders are not normally required to meet each other, let alone discuss new requirements and goals. In contrast, the new approach brings together the stakeholders in a series of meetings (stages 3.2, “Identify new served and serving systems (Stakeholder-Meeting-1)”, 3.3, “Identify goal conflict and outline new served and serving systems (Stakeholder-Meeting-2)”, and 3.4, “Resolve goal conflict and refine new served and serving systems (Stakeholder-Meeting-3)”). At these meetings the requirements engineer actively facilitates communication between the stakeholders as he or she prompts them to discuss goals, tradeoffs between goals, designs that satisfy goals, the impact of each design on each stakeholder, and so on.

And third, perhaps due to the first two problems, stakeholders often resist new systems [Fly98, KS98]. However, in the new approach, because there is a high level of involvement of stakeholders—particularly high in part-three, where both goals for new served and serving systems, and the mechanisms to achieve them are agreed—the chances are low of stakeholders subsequently resisting a system built to meet requirements that they have been closely involved in deriving.

A third group of problems is associated both with poor understanding of users’ work, and with the wrong problem often being solved. First, users’ work is often misunderstood [Mac96, LHG94], however the new approach explicitly investigates the work carried out by the client and the client’s subordinates in part-one of the new approach, “Conceptualise the current served system”. The investigations occur in steps 1.1.2, “Elicit and model the client’s view of the served system”, 1.3.2, “Elicit and model each subsystem owner’s view of their subsystem”, and 1.3.4, “Observe and interview served system workers at work”. These investigations should lead to a good understanding of the nature of the users’ work.

It is likely that the first problem often leads to a second: the wrong problem is solved [Fly98]. The new approach addresses this problem in the following way. Some of the early stages in part-one of the new approach are associated explicitly with eliciting the details of problems in the client’s domain and with prioritising them. Then in step 1.1.4, “Obtain the client’s agreement on the focused served system”, the client and the requirements engineer work together to agree the area(s) on which subsequent investigation will focus. Thus it can be seen that the method requires considerable attention to be focused by the requirements engineer and client upon identifying the most appropriate problem to solve. Because of this, the likelihood that the wrong problem is solved is reduced.

A fourth group of problems is associated with the effects and effectiveness of new computer-based systems. New systems are often associated with unwanted (negative) side-effects. But in the new approach, stage 3.3, “Iden-
identify goal conflict and outline new served and serving systems (Stakeholder-Meeting-2)", explicitly requires a requirements engineer to predict the consequences of any new design proposed by the stakeholders, in addition to those consequences actually desired. In other words, the requirements engineer is asked to predict unlooked for side-effects. The requirements engineer records such side-effects in a report, which the stakeholders are required to read and reflect upon. This mechanism is likely to reduce the incidence of negative consequences of a new system from occurring. In addition to reducing the incidence of negative consequences of a new system, the enactment of stage 3.3 of the new approach, “Identify goal conflict and outline new served and serving systems (Stakeholder-Meeting-2)”, by the requirements engineer and the stakeholders is also likely to reduce the incidence of ineffective systems. This is because, over and above identifying and reflecting upon unlooked for side-effects, the main purpose of stage 3.3 is for the requirements engineer and stakeholders to identify and reflect upon mechanisms for satisfying the desired goals. This process of identification and reflection is likely to reduce or eliminate the incidence of ineffective systems being created.

A fifth group of problems is associated with the approaches to requirements engineering from which the new approach was synthesised. First, as has been pointed out, descriptions of some socio-technical and goal-oriented approaches are incomplete. However, the new approach moves from conceptualising a current served system through to deriving requirements for new served and serving systems; and these two points delimit the scope of requirements engineering activities. In addition, all the stages and steps along the way are described. Thus it is considered that the new approach is complete. A second, related problem is that some parts of some of the approaches are not described in sufficient detail. However, the new approach is described in terms of its major parts, each part’s stages, and the steps associated with each stage. At the level of a step, a textual description that presents a step’s activity is given. Models in the approach are described using the UML’s extension mechanisms, and the approach’s tables and their allowed content are described fully. This level of presentation of detail should be sufficient to allow it to be applied by requirements engineering practitioners. A third, and again related, problem is that the notations for models used in some approaches lack adequate syntax and semantics. However, the models used in the new approach are presented using the UML’s extension mechanisms. Each one is presented with a class diagram, an iconography that shows, for example, the icon that represents each class, and an example of the model. It is considered that this constitutes an improvement over the representation of models in, for example, the Contextual Design approach [HB96, HB98]. Finally in this context, a problem for some approaches is their use of formal notation; this makes them difficult for users without the appropriate background in formal notations to use. However, the new approach does not involve the use of formal notation; thus it is considered easier to use than approaches that do.

A final problem is associated with the relationship between goals held by different stakeholders and/or different groups of stakeholders in an organisation. However, unlike some approaches, the new approach does not assume that a client’s perception of his or her goals will be the same as the client’s manager’s perception of the client’s goals. Nor does it assume that
the client’s subordinates’ perception of their goals will be the same as the client’s perception of their goals. Instead, it explicitly requires a requirements engineer to discover these three sets of goals independently. In doing so, the approach avoids the dubious assumption that the goals of an enterprise (or client) correspond to the goals of the members of the enterprise (or client’s subordinates).

This review shows that most of the problems of other related approaches—traditional, socio-technical, and goal-oriented approaches—are likely to be attenuated or eliminated in the new approach. However, it should also be noted that the approach could be modified to improve its treatment both of an organisation’s high-level objectives and business needs, and also of the wider organisational context of a client’s domain.

8.3 Evaluation of the contribution of the thesis

Chapter One describes the contribution of this thesis as follows. First, a framework of principle and features for guiding the design of requirements engineering methods has been developed from the results of critically reviewing some established socio-technical, goal-oriented, and process-oriented approaches to requirements engineering. Second, the framework of principles and features has been used to synthesise a new goal-oriented approach to requirements engineering.

Third, the new approach is sufficiently articulated to allow its use to derive requirements for computer-based systems that will support organisational activities. And fourth, for a given organisational context, one outcome of applying the approach is an indication of what computer-based systems to build out of all those that might be built. This section evaluates the extent to which these claims can be substantiated. In addition, it looks at the costs associated with the synthesised approach, and suggests improvements that might be made to it.

For the first claim, the beginning of Chapter Six uses the results of the critical reviews of some established socio-technical, goal-oriented, and process-oriented approaches to establish a framework of principles and features for both guiding the development of requirements engineering methods, and for evaluating existing methods. And the existence of this framework substantiates the first claim.

For the second claim, Chapter Six presents an approach to requirements engineering that has resulted from applying the principles, and using the features, of the framework. So the second claim is also substantiated.

For the third claim, the approach has been applied in one case study and has generated a set of requirements for a computer-based system which, if implemented, would arguably support the associated organisational activities (see Chapter Seven). And, because of the goal-oriented, process-oriented, and participative nature of the new approach, it is predicted that the requirements, and thus the implementation, would be of a higher quality than requirements (and thus implementations) produced by other methods - higher quality in the sense that the goals and work of all stakeholders would be better supported. So it would seem that the third claim has received some support. In order to assess the exact amount of support, it is necessary to consider any special characteristics of this case study. The special
CHAPTER 8. CRITICAL EVALUATION

characteristics include the following:

- The approach was applied by its designer, who acted as a requirements engineer
- The approach was applied in a case study within a University
- There were few, if any, differences of opinion about important goals among the stakeholders involved in the case study—the client, the dean, the computer officers, and the staff and student users
- At the end of the case study the stakeholders were dominated by a new client

First, the approach was applied by its designer, who acted as a requirements engineer. So the approach was applied by an expert who was already committed to trying to ensure that it would work. It still remains to be seen whether, and to what extent, other requirements engineering practitioners could understand and apply the approach. Second, the approach was applied in a case study within a University. Most of those involved in the case study either had, or were on the way to obtaining, higher-education qualifications: HNCs, HNDs, degrees, MScs, and PhDs. As such, this group is likely to be more highly educated and motivated than groups in other kinds of organisations. So it still remains to be tested whether, and to what extent, the lower-level of education and commitment likely to be found in many other kinds of organisation would impact the effectiveness of the approach. Third, in the case study it transpired that there were few, if any, major differences of opinion over important goals among the stakeholders. However, in many organisations or parts of organisations it is quite likely that significant differences would be found over what should be the important goals. It remains to be tested whether, and to what extent, application of the approach in such circumstances would yield helpful results, in other words, would yield requirements for computer-based systems that could be used to support organisational activities. Finally, towards the end of the case study a new client joined the group of stakeholders. This client dominated the decision-making process in the formal meetings of stakeholders. It is not clear to what extent the existence of such a dominant voice reduces the effectiveness of the approach by leading to a reduction in the utility of the requirements it produces. On the one hand, a dominant stakeholder might be able to ensure that users did use any computer-based systems developed from requirements for which he or she was mainly responsible. On the other hand, it may be the case that the use of such computer-based systems might decline over time exactly because they were not based upon authentic accommodations among stakeholders during the requirements engineering process.

For the fourth claim, when the approach is applied, how helpful is it in indicating which computer-based systems should be built out of all those that might be built? It can be argued that the approach is very effective here. Application of part-one of the approach in the case study surfaced a wide-range of activities and concerns from those involved in or affected by the domain, from user-problem management to the selling of a variety of materials. In addition, the approach encouraged the client to consider all views before prioritising those areas where further work was to be focused.
These features—uncovering the full range of current activities, ensuring that the client reflects upon it all, and requiring the client to prioritise areas to focus upon—are arguably likely to lead to more appropriate computer-based systems being selected to be built.

8.4 Summary

This chapter has both evaluated the new synthesised goal-oriented approach, and assessed the extent to which the claims made for it in the first chapter have been met. The approach was evaluated in two ways. First, the framework of principles and features, developed earlier, was used to assess it, and the result was compared to similar assessments made of selected established approaches. The comparison indicated that the new synthesised goal-oriented approach was seen to be superior in its design and development. Second, an assessment was made of the extent to which the new synthesised goal-oriented approach addressed some key problems in requirements engineering that had been identified earlier. The new approach was considered to address these problems effectively.

With regard to assessing the claims made for the new approach, it was concluded that a framework of principles and features had been created, that it had been used to synthesise a new goal-oriented approach to requirements engineering, and that the new approach would be effective in indicating which CBSs to build out of all those that might be built in a given context. However, because of the limitations of the case study approach, the full range of applicability of the new approach remains to be determined.

The final chapter of the thesis summarises the work undertaken, highlights the main achievements, and outlines future work.
Chapter 9

Conclusion
9.1 Introduction

This chapter first summarises the work undertaken in support of this thesis. It then highlights the main achievements of the work. The final section considers future work to enhance and further validate the new synthesised approach.

9.2 Thesis summary

The thesis begins by describing some problems associated with early approaches to requirements engineering. Three socio-technical approaches are described that were intended to tackle these problems: Checkland’s Information System Development, [WBPB95], Contextual Design [HB96, HB98], and Eason’s socio-technical approach [Eas88]. Arguably the principles, techniques, and ideas embodied in these approaches should make them effective in tackling the problems. However, it is not yet clear how effective they are. One reason for this may be that they have not been widely applied in real-world contexts, and one reason for this may be that there are insufficiently detailed descriptions of how to apply them.

Another group of more recent approaches to requirements engineering is also reviewed: the goal-oriented approaches, which include, for example, KAOS [DvF93] and Loucopoulos’s approach [LK95a]. Some of these, for example KAOS, are considered to be difficult for the non-expert to apply, while others, for example Loucopoulos’s approach, are considered to be described in insufficient detail to allow them to be taken up and applied by the non-expert.

To tackle the problems described above, a framework of principles and features for guiding the development of requirements engineering methods was derived from a critical review of the approaches considered during the literature review. This framework was used to synthesise a new goal-oriented approach to requirements engineering. And to show that the new approach was viable, it was applied to a real-world case study.

9.3 Main achievements

The two main achievements of the work undertaken in support of this thesis have been as follows. First, a framework of principles and features for guiding the design and development of requirements engineering methods has been established. Second, the framework has been used both to synthesise a new goal-oriented approach to requirements engineering, and to compare it to some established approaches.

The new approach is described in significantly greater detail than the approaches that it is based upon. This should encourage its widespread application to real-world problems by requirements engineering practitioners. If this occurs, it will then be possible to evaluate the effectiveness of goal-based approaches to requirements engineering. A clearer idea about which contexts are most suitable for treatment with a goal-oriented approach will also emerge. And it will be possible to compare goal-oriented approaches to other recent approaches like use-case modelling [Jac93] or viewpoint modelling [KS96, FKNG92].
The new approach embodies some of the principles, ideas, and techniques of earlier approaches. For example, it incorporates the idea that the served system should be understood before the serving system is conceptualised [WBPB95]. So whenever the new approach is used, it will be possible to assess the effectiveness of those principles, ideas, and techniques in solving the problems of early approaches to requirements engineering—poor domain knowledge, poor stakeholder communication, and so on. Thus, significantly, it will be possible to assess how well the new approach—and by implication the principles, techniques and ideas that it embodies—facilitates both the selection of which computer-based systems to develop (from all those that might be developed) [WBPB95], and the determination of what functionality should be included in the selected computer-based systems. This may be viewed as an additional achievement of the work undertaken to support this thesis.

In addition to the main contributions for this thesis, the new synthesised goal-oriented approach is also associated with some important features, which are worth emphasising here. First, approaches to requirements engineering typically start with a pre-existing, informal statement of requirements and proceed by identifying, expressing, formalising, and organising them. But, in practice, deriving the first statement of requirements is difficult; this is in part because there are no established methods for doing it. The new, synthesised approach provides just such a method for deriving a statement of requirements. In doing this, it adds to the utility of requirements engineering.

Second, although some requirements engineering approaches have considered domain problems as a source of requirements, none have done so to a significant extent. But domain problems are a key indicator of areas where improvements can be made in a domain. The new, synthesised approach emphasises the identification and use of domain problems as key drivers of improvements in a domain; in particular, domain problems are used for determining requirements for associated supporting computer-based systems in the domain.

Third, established goal-oriented approaches to requirements engineering, tend to capture and use just the current goals of a domain for deriving requirements for computer-based systems that will support the domain. But, in doing this, these approaches miss the opportunity to improve the domain by capturing and then supporting new goals. The new, synthesised approach captures both current goals and new goals, and, in doing so, opens up the possibility of enabling more improvement to a domain to be achieved.

Fourth, unlike other prominent goal-oriented approaches, the new, synthesised approach requires the stakeholders themselves to be actively involved in enacting substantial parts of the approach: for example, they are involved in establishing current and new goals for a domain, and in generating incremental and radical ideas for achieving agreed goals. By doing this, the new approach not only avoids the problems associated with not involving stakeholders—rejection of subsequent computer-based systems, for example—but also taps a knowledgeable source of ideas for both goals and the mechanisms to achieve them.

Fifth, unlike other approaches to requirements engineering, the new, synthesised approach encourages stakeholders to fully appreciate the full
impact of their ideas for satisfying goals. This is done by getting them to predict additional effects (side effects) of their ideas, and then evaluating those effects—positive, neutral, or negative—with respect to each stakeholder. The results are intended to be used to evaluate alternative suggestions for satisfying goals. This feature of the new, synthesised approach should help to weed out computer-based systems that, while satisfying new, agreed goals, unfortunately produce unwanted side-effects.

Sixth, traditional goal-oriented approaches require a requirements engineer to elicit high-level goals, and the decompose them to create a goal-hierarchy. The hierarchy is used to derive requirements for computer-based systems. This method can lead to the identification and use of lower-level goals that do not correspond to the reality of a domain, and thus to the identification of requirements for inappropriate computer-based systems. By contrast, the new, synthesised approach elicits goals which may be at a range of levels, from high to low. The approach builds these into a goal-hierarchy, adding “missing” goals if necessary. In doing this, the new approach leads to the identification of requirements for systems that meet goals that correspond to stakeholders’ actual goals.

Seventh, although other approaches to requirements engineering do often feature validation, the new, synthesised goal-oriented approach features it more prominently: stakeholders are required to critically review each important output from the approach. This means that the end result—requirements for computer-based systems—is highly likely to satisfy the client and other stakeholders.

Finally, this work constitutes an attempt to synthesise (and validate) an approach from selected features of three families of approaches to requirements engineering: socio-technical, goal-oriented, and process-oriented. And synthesising an approach from these three particular families of approaches has not been tried before.

9.4 Future work

This section considers future work. The new approach compared well to established approaches, and is considered to be likely to produce sets of high quality requirements. Nevertheless it might be further enhanced if some of the following work was undertaken. First, a set of tools could be designed and built that would support the approach. For example, the set might include tools designed to provide active method guidance to users, and tools that ensured that only well-formed models could be eventually passed as complete. In addition, it would be useful to have a tool that supported users in the creation of goal hierarchies and composite goal hierarchies. Tools like these would be likely to facilitate both the correct application of the approach, and the production of well-formed models, as well as reduce the overall time of application.

The overall time of application might also be reduced if work were directed towards simplifying the approach and increasing its flexibility. The former might be achieved by, for example, simplifying stages and steps by omitting less critical activities. Increasing flexibility of application might be achieved through investigating whether and to what extent, different parts, stages, and steps of the approach might be applied out of the currently
recommended sequence.

As well as further work that might enhance the new approach, it would be useful to further validate it by testing the approach across a range of organisation types and with requirements engineering practitioners of differing levels of experience with it. In all cases, the utility, usability, and actual use of CBSs created using the new approach should be evaluated as part of the validation.

Recent-developments-driven work

In addition to future work that is intended to enhance or further validate the new approach, some future work could be based upon the results of recent research in related areas. Research into goal-oriented approaches to requirements engineering has continued over the period during which this thesis was written. Part of the work following on from the thesis could involve making a detailed assessment of how the results of the recent work may be incorporated into the new approach. The recent research is now presented.

KAOS:

Van Lamsweerde and his colleagues continue to refine KAOS [DvF93], a very formal approach to requirements engineering (see Chapter Four). Recent work on KAOS has encompassed the following areas:

- The refinement of goals and assignment of agents using a catalogue of tactics [LvL02a]
- The provision of techniques for operationalising system goals [LvL02b]
- The provision of techniques and heuristics for detecting and resolving goal conflict [vLDL98]
- The provision of techniques for identifying and resolving obstacles—in other words anticipated exceptional system behaviour—implicit in elicited goals and requirements [vLL00]

The work on the refinement of goals may be relevant to the inferencing of goals by a requirements engineer, an activity of the new approach. The work on operationalising system goals may be relevant to ensuring, in the new approach, that sufficient operational goals have been identified. And the work on conflict may be relevant to the creation of Stakeholder Goal Hierarchy Diagrams and Composite Goal Hierarchy Diagrams.

Goals and processes:

A number of independent research initiatives have focused in different ways upon the relationship between goals and processes, which is also a concern of this thesis. For example, Loucopoulos presents a goal-driven approach to business modelling in order to support “closer alignment between intensional and operational aspects of the organisation” [KL99]. In a related way, Kawalek and Kueng have created a design method for modelling a business process from its functional goals [KK97]. Kawalek et al. have also explored
how a combination of goal-modelling and process-modelling techniques may be used to facilitate organisational learning [KWL+00]. In another example of similar research, Downs and Lunn describe an approach to analysing a process support system [DL02]. Their approach requires an analyst to perform the following activities:

- State the vision and goals of a business
- Analyse and assess the current business processes
- Create a revised business process description

This approach is very similar to parts of the new approach, presented above.

In addition to these approaches, a substantial, collaborative effort is being made to produce a requirements-driven software development method called Tropos. And this also focuses upon goals and processes. (See, for example: [CKM02, GMNS02].) The Tropos method has four stages:

- Study an organisational setting to understand the problems
- Define the system-to-be within its operational environment
- Create the architectural design
- Create the detailed design

While stages three and four do not have a counterpart in the new approach, stage-one is similar to the new approach in its emphasis on understanding a current served system, and stage-two is also similar: the “system-to-be” has its counterpart in a new serving system, and the “operational environment”, in the new served system.

This research into the relationship between goals and processes needs to be critically reviewed to determine whether any results can be usefully exploited for the new, synthesised approach.
Bibliography


[KWL+00] P. Kawalek, D. Wastell, E. Locke, M. Willetts, and P. Kueng. Seeking the goal in the process, the process for the goal: Organisational


BIBLIOGRAPHY


Appendices
Appendix A

The Meeting Scheduler
Problem: Preliminary Definition

Problem Statement

The purpose of a meeting scheduler is to support the organization of meetings - that is, to determine, for each meeting request, a meeting date and location so that most of the intended participants will effectively participate. The meeting date and location should thus be as convenient as possible to all participants. Information about the meeting should also be made available as early as possible to all potential participants. The intended system should considerably reduce the amount of overhead usually incurred in organizing meetings where potential attendees are distributed over many different places.

Meetings are typically arranged in the following way. A meeting initiator asks all potential meeting attendees for the following information based on their personal agenda:

- a set of dates on which they cannot attend the meeting (hereafter referred as exclusion set);
- a set of dates on which they would prefer the meeting to take place (hereafter referred as preference set).

A meeting date is defined by a pair (calendar date, time period). The exclusion and preference sets are contained in some time interval prescribed by the meeting initiator (hereafter referred as date range).

The initiator also asks active participants to provide any special equipment requirements on the meeting location (e.g., overhead-projector, workstation, network connection, telephones, etc.); he/she may also ask important participants to state preferences about the meeting location.

The proposed meeting date should belong to the stated date range and to none of the exclusion sets; furthermore it should ideally belong to as many preference sets as possible. A date conflict occurs when no such date can be found. A conflict is strong when no date can be found within the date range and outside all exclusion sets; it is weak when dates can be found within the
date range and outside all exclusion sets, but no date can be found at the
intersection of all preference sets. Conflicts can be resolved in several ways:

- the initiator extends the date range;
- some participants remove some dates from their exclusion set;
- some participants withdraw from the meeting;
- some participants add some new dates to their preference set.

A meeting room must be available at the selected meeting date. It should
meet the equipment requirements; furthermore it should ideally belong to
one of the locations preferred by as many important participants as possible.
A new round of negotiation may be required when no such room can be
found.

The meeting initiator can be one of the participants or some representative (e.g., a secretary).

The system should assist users in the following activities.

- Plan meetings under the constraints expressed by participants (see
above).

- Replan a meeting dynamically to support as much flexibility as possi-
ble. On one hand, participants should be allowed to modify their
exclusion set, preference set and/or preferred location before a meeting
date/location is proposed. On the other hand, it should be possible to
take external constraints into account after a date and location have
been proposed - e.g., due to the need to accommodate a more impor-
tant meeting. The original meeting date or location may then need to
be changed; sometimes the meeting may even be cancelled.

- Support conflict resolution according to resolution policies stated by
the client.

- Manage all interactions among participants required during the orga-
nization of the meeting - to communicate requests, to get replies even
from participants not reacting promptly, to support the negotiation
and conflict resolution processes, to make participants aware of what’s
going on during the planning process, to keep participants informed
about schedules and their changes, to make them confident about the
reliability of the communications, etc.

The meeting scheduler must in general handle several meeting requests
in parallel. Meeting requests can be competing by overlapping in time or
space. Concurrency must thus be managed.

The following aspects should also be taken into account.

- The system should accommodate decentralized requests; any authorized
user should be able to request a meeting independently of his whereabouts.

- Physical constraints may not be broken - e.g., a person may only attend
one meeting at a time, a meeting room may not be allocated to more
than one meeting at the same time, etc.
APPENDIX A. THE MEETING SCHEDULER PROBLEM:
PRELIMINARY DEFINITION

- The system should provide an appropriate level of performance, for example:
  - the elapsed time between the submission of a meeting request and the determination of the corresponding meeting date/location should be as small as possible;
  - the elapsed time between the determination of a meeting date/location and the communication of this information to all participants concerned should be as small as possible;
  - a minimal delay should exist between the determination of a meeting date and the actual date of the meeting.

- Privacy rules should be enforced; an ordinary participant should not be aware of constraints stated by other participants.

- The system should be usable by non-experts.

- The system should be customizable to professional as well as private meetings. These two modes of use are characterized by different restrictions on the time periods that may be allocated (e.g., meetings during office hours, private activities during leisure time).

- The system should be flexible enough to accommodate evolving data - e.g., the sets of concerned participants may be varying, the address at which a participant can be reached may be varying, etc.

- The system should be easily extendable to accommodate the following typical variations:
  - introduction of explicit status and priorities among participants;
  - introduction of explicit priorities among dates in preference sets;
  - introduction of explicit dependencies between meeting date and meeting location;
  - participation through delegation - a participant may ask another person to represent him/her at the meeting;
  - partial attendance - a participant can only attend part of the meeting;
  - variations in date formats, address formats, interface language, etc.
  - partial reuse in other contexts - e.g., to help establish course schedules.
Appendix B

Acronyms and glossary of terms

Table B.1: Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Expansion</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD</td>
<td>Affinity Diagram</td>
</tr>
<tr>
<td>ATC</td>
<td>Air Traffic Control</td>
</tr>
<tr>
<td>CBPMS</td>
<td>Computer Based problem Management System</td>
</tr>
<tr>
<td>CBS</td>
<td>Computer-based System</td>
</tr>
<tr>
<td>CD</td>
<td>Contextual Design</td>
</tr>
<tr>
<td>CI</td>
<td>Contextual Inquiry</td>
</tr>
<tr>
<td>CO</td>
<td>Computing Officer</td>
</tr>
<tr>
<td>COTS</td>
<td>Computer Officers and Technical Support</td>
</tr>
<tr>
<td>CWM</td>
<td>Consolidated Work Model</td>
</tr>
<tr>
<td>FAQ</td>
<td>Frequently Asked Questions</td>
</tr>
<tr>
<td>GOA</td>
<td>Goal-oriented Approach</td>
</tr>
<tr>
<td>GMARC</td>
<td>Generic Modelling and Requirements Capture</td>
</tr>
<tr>
<td>HAS</td>
<td>Human Activity System</td>
</tr>
<tr>
<td>ISAC</td>
<td>Information Systems Work and Analysis of Changes</td>
</tr>
<tr>
<td>ISD</td>
<td>Information System Development</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>KAOS</td>
<td>Knowledge Acquisition in Automated Specification of Software</td>
</tr>
<tr>
<td>RAD</td>
<td>Role Activity Diagram</td>
</tr>
<tr>
<td>RE</td>
<td>Requirements Engineer</td>
</tr>
<tr>
<td>SSM</td>
<td>Soft Systems Methodology</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modelling Language</td>
</tr>
</tbody>
</table>
Table B.2: Term definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>An action is a behaviour, of a person or agent, that acts upon the objects comprising a state; it may cause them to change. In other words, an action may change a state.</td>
</tr>
<tr>
<td>Agent</td>
<td>An agent is a human or non-human object that may perform actions.</td>
</tr>
<tr>
<td>Constraint</td>
<td>Some constraints are requirements—something either asked for or viewed as necessary. These constraints are a particular kind of requirement that place some limit along a dimension of the requirement. For example, a constraint may specify a maximum cost, or a maximum amount of memory available, and so on. Other constraints are immutable: they do not need to be requested, merely acknowledged. Examples of such constraints include gravity and the average attention span of humans. Some constraints fall in between these two kinds of constraints. Examples include legal constraints like the Data Protection Act, organisational standards, and organisational technical infrastructures. Such constraints may normally be treated as immutable, although, in theory, it is possible to change them. They are like requirements in that some person or agency ought to confirm conformance to them—a confirmation which would be like a request: for example, “the system new should conform to the current organisational technical infrastructure”.</td>
</tr>
<tr>
<td>Goal</td>
<td>A goal is a property, applicable to a part of the real world, that a person or agency desires to hold. The property may be satisfied by one or more states, or by one or more sequences of states. For example the light must be on in room 2P32 at the University of the West of England between 0900h and 1700h on Monday 3 November.</td>
</tr>
<tr>
<td>Leaf-goal</td>
<td>A leaf-goal is a goal with no dependent sub-goals.</td>
</tr>
<tr>
<td>Problem</td>
<td>A problem is a recurring state or sequence of states associated with one or more properties that are deemed by a person or agency to be undesirable. For example, users of a particular service may describe one problem with the service as being that their service requests are often lost. A problem may normally be transformed into a corresponding goal. For example, if the problem is “The red light is on when it should not be on”, the corresponding goal might be “Ensure that the red light is on only when it should be on”.</td>
</tr>
<tr>
<td>Process</td>
<td>A process comprises two or more activities, performed by people or agencies, in sequence and/or in parallel, usually in prespecified ways, in order to achieve a predetermined goal.</td>
</tr>
<tr>
<td>Requirement</td>
<td>A requirement is either something that some person or agency has requested, or something that a person or agency views as necessary. The “something” expresses a property (i.e. goal) of the real world: it could be a property of one or more systems-as-are, or of systems-to-be, or of environments-as-are, or of environments-to-be. For example, in some hypothetical case, the user stakeholders have requested that the new control system provide a help facility.</td>
</tr>
<tr>
<td>Satisfice</td>
<td>To satisfice is to partially satisfy a goal.</td>
</tr>
<tr>
<td>State</td>
<td>A state is a configuration of some part of the world (place) at some point in time, in terms of its constituent objects and their attributes and values. For example, in room 2P32 at the University of the West of England, on the 3rd of November, 2003, there are two desks, two PCs, four chairs, and so on.</td>
</tr>
<tr>
<td>State boundary</td>
<td>State boundaries are defined by people. For example, a state boundary may be deemed by a person or agency to occur every second, or on every hour, or whenever a particular attribute value of a particular object changes.</td>
</tr>
</tbody>
</table>
Appendix C

RAD notation
APPENDIX C. RAD NOTATION

Figure C.1: The RAD notation
Appendix D

Client questionnaire pro-forma
Questionnaire

1. High-level goals:

What do you consider to be the main high-level goals in your domain of responsibility? For each goal can you supply the following information:

(a) A name and description for the goal.
(b) The name of the person or agency that assigned the goal to you, and the capacity in which they were acting.
(c) The kind of goal this is.
(d) The importance of this goal.

Please use copies of form Form-1 to supply this information.

2. Stability of goals:

How frequently do your goals change (tick (a) or (b)):

(a) Goals in my domain of responsibility change frequently.
(b) Goals in my domain of responsibility are relatively stable.

3. Domain structure:

Can you describe the high-level structure of your domain of responsibility. For each sub-division in your domain of responsibility can you supply the following information:

(a) The name of the sub-division.
(b) The sub-division leader’s name and their role name.

(c) Goals that you have assigned to the sub-division.
   For each goal, can you supply the following information:
   
   o Goal description
   o Goal type
   o Goal importance

(d) The nature of the sub-division’ work

Please use copies of form Form-2 to supply this information.

4. Domain tasks:
Can you describe the main high-level tasks that you carry out in your domain of responsibility. For each task in your domain of responsibility can you supply the following information:

(a) A name and description for the task.

(b) The task’s importance.

Please use copies of form Form-9 to supply this information.

5. Planned major changes:

Can you outline any major changes that you have planned for your domain of responsibility.

6. Common Goals:

Are there any goals which you have assigned in common to all of your sub-divisions.

Please use copies of form Form-4 to supply this information.

7. Problems:

What do you perceive to be the main problem areas within your domain of responsibility? For each problem area, can you supply the following information:

(a) A name and description for the problem.

(b) The severity of the problem.

(c) The frequency of occurrence of the problem.

(d) The goal(s) that the problem impacts the most.

Please use copies of form Form-5 provided to supply this information.

8. Improvements

What aspect of your domain of responsibility would you like to improve (apart, that is, from eliminating the problems you have described above)? For each aspect could you supply the following information:

(a) The improvement that you would like (preferably quantified).

(b) The goals that the improvement would impact.
(c) The importance of the improvement.

Please use copies of form Form-7 to supply this information.

9. Improving the questionnaire

Do you think that the information you have supplied here would allow the questionnaire analyst to build up a comprehensive picture of both the context of your domain of responsibility and of its main problem areas. If not, can you indicate below what additional aspects you feel ought to be covered.

10. Explanatory notes:

Which questions would benefit from having additional explanatory notes?

11. Time spent:

How long did it take you to complete this questionnaire?
APPENDIX D. CLIENT QUESTIONNAIRE PRO-FORMA

Forms

Form-1

Form-1: High-level goal:
------------------------

(a) Goal name:

Goal description:

(b) Source of goal (indicate name and role):

(c) Goal type (tick up to two options):

- o increase  o avoid
- o maximise  o cease from
- o achieve   o minimise
- o maintain  o decrease

(d) Importance of goal (tick one option):

- o Highest importance
- o Important
- o Not very important

------------------------
Form-2

Form-2: Sub-division:
---------------------

(a) Sub-division name:

(b) Sub-division leader’s name and role name:

(c) Goals you have assigned to the sub-division:

Please use copies of form Form-3 to describe sub-division goals.

Form-3

Form-3: Sub-division goal:
--------------------------

(a) Sub-division name:

(b) Goal name:

Goal description:

(c) Goal type (tick up to two options):

   o increase               o avoid
   o maximise               o cease from
   o achieve                o minimise
   o maintain               o decrease

(d) Importance of goal (tick one option):

   o Highest importance
   o Important
   o Not very important

(a) Sub-division name:

(b) Goal name:

Goal description:

(c) Goal type (tick up to two options):

   o increase               o avoid
APPENDIX D. CLIENT QUESTIONNAIRE PRO-FORMA

(d) Importance of goal (tick one option):

- Highest importance
- Important
- Not very important

Form-4

Form-4: Common goal:
--------------

(a) Goal name:

Goal description:

(b) Goal type (tick up to two options):

- Increase
- Avoid
- Maximise
- Cease from
- Achieve
- Minimise
- Maintain
- Decrease

(c) Importance of goal (tick one option):

- Highest importance
- Important
- Not very important

--------------------------------------------------------------------------------

(a) Goal name:

Goal description:

(b) Goal type (tick up to two options):

- Increase
- Avoid
- Maximise
- Cease from
- Achieve
- Minimise
- Maintain
- Decrease

(c) Importance of goal (tick one option):

- Highest importance
- Important
APPENDIX D. CLIENT QUESTIONNAIRE PRO-FORMA

- Not very important

Form-5

Form-5: Problems:
-----------------

(a) Problem name:

Problem description:

(b) Problem severity (tick one option):

- Extremely severe
- Very severe
- Severe
- Not very severe
- Mild

(c) Frequency of occurrence of problem (tick one option):

- Continuous
- Weekly
- Hourly
- Daily
- Monthly
- Yearly
- Other
- (please specify)

(d) Impacted goals:

Please use Form-6 to supply this information.

--------------------------------------------------------------

(a) Problem name:

Problem description:

(b) Problem severity (tick one option):

- Extremely severe
- Very severe
- Severe
- Not very severe
- Mild

(c) Frequency of occurrence of problem (tick one option):

- Continuous
- Weekly
- Hourly
- Daily
- Monthly
- Yearly
- Other
- (please specify)

(d) Impacted goals:
Please use Form-6 to supply this information.

**Form-6**

**Form-6: Impacted goals:**

(a) Problem name:

(b)

Goal name:

Goal description:

Nature of impact:

(b)

Goal name:

Goal description:

Nature of impact:

(b)

Goal name:

Goal description:

Nature of impact:

**Form-7**

**Form-7: Improvements:**

(a) Improvement name:

Description of improvement:

(b) Importance of this improvement:

- Extremely important
- Very important
- Important
- Not very important
APPENDIX D. CLIENT QUESTIONNAIRE PRO-FORMA

o Mildly important

(c) Impacted goals:

Please use copies of form Form-8 to supply this information

---------------------------------------------------------------

(a) Improvement name:

Description of improvement:

(b) Importance of this improvement:

  o Extremely important
  o Very important
  o Important
  o Not very important
  o Mildly important

(c) Impacted goals:

Please use copies of form Form-8 to supply this information

Form-8

Form-8: Impacted goals:
------------------------

(a) Improvement name:

(b)

  Goal name:

  Goal description:

  Nature of impact:

(b)

  Goal name:

  Goal description:

  Nature of impact:

(b)

  Goal name:

  Goal description:
Nature of impact:

Form-9

Form-9: High-level tasks:
-------------------------

(a) Task name:

Task description:

(b) Task importance:

  o Very important
  o Important
  o Note very important

(a) Task name:

Task description:

(b) Task importance:

  o Very important
  o Important
  o Note very important

(a) Task name:

Task description:

(b) Task importance:

  o Very important
  o Important
  o Note very important
Notes of guidance for completing the questionnaire

1.

(a)

A goal is a description of a state of affairs to be achieved or maintained, etc. For an example of a goal to be achieved consider the following:

"To increase company profit."

In general, goals are more useful when they are described in clear rather than in vague terms, and, in particular, when they are described quantitatively. For example, the goal above might be improved to the following:

"To increase company profit by 50% within 2 years from 1/11/96."

(b)

One source of goals might be your boss, Joe Smith, so you would write:

Joe Smith / Boss

Another source might be a committee, for example a steering committee, so you would just write:

Steering committee

You may have generated one or more goals yourself in which case you should just write:

Self-generated

(c)

A goal may be achieved. For example the goal discussed above is one which may be achieved. A goal may also be maintained. For example, "To provide a quality service to customers" is a goal which would normally be maintained. There are other types of goals including the following: increase, maximise, achieve, maintain, avoid, cease from, minimise, and decrease.

(d)

Self-explanatory.
2.

In some organisations the main goals change frequently. For example, currently (1996) many businesses perceive that to remain competitive they must frequently target new goals. On the other hand in some organisations the main goals remain the same year in year out.

3.

A domain may be structured into sub-divisions. For example, companies are often divided into divisions. Again company divisions are often divided into functional areas like marketing or sales. Or again, a project may be divided into teams.

(a)

A short descriptive name is required here. For example "plastics division", or "marketing", or "user interface team", etc.

(b)

Examples might include:

J. Davis / Team leader
T. Smith / Head of department

(c)

You should include here the goals that you have assigned to this sub-division. Goals assigned in common to all sub-divisions are collected later.

(d)

You should indicate here the main tasks carried out by the sub-division.

4.

Self-explanatory.

5.

Self-explanatory.
6. You should indicate here all the goals that you have assigned in common to all the sub-divisions.

7. (a) Here, you should describe the problem, including, for example, when it occurs, who is involved, what are the undesirable consequences, etc.

(b), (c), (d) Self-explanatory.

8. Self-explanatory.


10. Self-explanatory.

Appendix E

Subsystem owner questionnaire pro-forma
Generic questionnaire for subsystem owners

The generic questionnaire for subsystem owners is given below. This should be specialised for each new project with questions designed to elicit the subsystem owners’ views of the problem(s) identified by the client. For the case study reported in this thesis two such modified questionnaires were created: one for the Helpdesk Manager and one for the other subsystem managers—e.g. Unix System Manager, etc. These two complete questionnaires are shown below:

1. High-level goals:

As a subsystem manager, what do you consider are the high-level goals of your domain of responsibility? For each goal, can you supply the following information:

(a) A name and description for the goal.

(b) The name of the person or agency that assigned the goal to you, and the capacity in which they were acting. (This may include self-assigned goals.)

(c) The kind of goal this is. (Increase, Maximise, Achieve, Maintain, Avoid, Cease from, Minimise, Decrease.)

(d) The importance of this goal. (Highest Importance, Important, Not Highest Importance.)

2. Stability of goals:

How frequently do your goals change?

- Goals in my domain of responsibility change frequently.
- Goals in my domain of responsibility are relatively stable.

Questionnaire for the Helpdesk Manager

1. High-level goals:

As the Helpdesk Manager what do you consider are the main high-level goals of your domain of responsibility, the Helpdesk? For each goal can you supply the following information:

(a) A name and description for the goal.

(b) The name of the person or agency that assigned the goal to you, and the capacity in which they were acting. (This may include self-assigned goals)

(c) The kind of goal this is. (Increase, Maximize, Achieve, Maintain, Avoid, Cease from, Minimize, Decrease)

(d) The importance of this goal. (Highest Importance, Important, Not Highest Importance.)
2. Stability of goals:

How frequently do your goals change (tick one of the following):

(a) Goals in my domain of responsibility change frequently.
(b) Goals in my domain of responsibility are relatively stable.

3. Do you consider that resolving staff problems, requests for advice, requests for information, etc. is one of the goals of the Helpdesk? (Please tick one option)

(i) Yes
(ii) No

4. If you answered “yes” to the previous question, could you indicate how important the goal is to the Helpdesk. (Please tick one option)

(i) Not very important
(ii) Important
(iii) Highest importance

5. (a) What percentage of user problems, advice or information requests, etc. do you resolve on the spot while the user is with you? (Write down a percentage from 0% to 100%)

(b) What percentage of user problems, advice or information requests, etc. do you pass on to another person or group? (Write down a percentage from 0% to 100%)

(c) What percentage of user problems, advice or information requests, etc. do you add to your own list of problems to be investigated and resolved? (Write down a percentage from 0% to 100%)

6. What are the main problems that you experience in managing the help desk? In particular what are the main problems that you experience in managing user problem management?

7. In an ideal world, how would you like to improve the Helpdesk problem management system?

8. Is there anything else that you think I should know about problem management?

Questionnaire for the other subsystem owners

1. High-level goals:

As the subdivision System Manager what do you consider are the main high-level goals of your domain of responsibility, the Helpdesk? For each goal can you supply the following information:

(a) A name and description for the goal.
APPENDIX E. SUBSYSTEM OWNER QUESTIONNAIRE

PRO-FORMA

(b) The name of the person or agency that assigned the goal to you, and the capacity in which they were acting. (This may include self-assigned goals)

c) The kind of goal this is. (Increase, Maximize, Achieve, Maintain, Avoid, Cease from, Minimize, Decrease)

d) The importance of this goal. Highest Importance, Important, Not Highest Importance

2. Stability of goals:

How frequently do your goals change (tick one of the following):

(a) Goals in my domain of responsibility change frequently.
(b) Goals in my domain of responsibility are relatively stable.

3. Do you consider that resolving staff problems, requests for advice, requests for information, etc. is one of the goals of your subdivision? (Please tick one option)

(i) Yes
(ii) No

4. If you answered “yes” to the previous question, could you indicate how important the goal is to your subdivision. (Please tick one option)

(i) Not very important
(ii) Important
(iii) Highest importance

5. (a) What percentage of user problems, advice or information requests, etc. do you resolve on the spot while the user is with you? (Write down a percentage from 0)
(b) What percentage of user problems, advice or information requests, etc. do you pass on to another person or group? (Write down a percentage from 0)
(c) What percentage of user problems, advice or information requests, etc. do you add to your own list of problems to be investigated and resolved? (Write down a percentage from 0)

6. I would like you now to consider how you manage those problems, etc. which you have added to your own list of problems to be investigated and resolved.

(a) Do you record the existence of the problem in some way e.g. writing down the details or do you rely on your memory? (Please tick one option)

(i) Record problem details (specify how)
(ii) Rely on memory to retain problem details

(b) If you have a list of longterm problems to investigate and resolve, how do you decide in what order to tackle them? (E.g. do you assign a priority to them either explicitly or in your head?)
(c) On average, how long is your list of long-term problems?

(d) What happens to problems that do not seem possible to resolve?

(e) Do you to keep the user (problem owner) informed of the current status of a problem?
   (i) yes
   (ii) no

(e) If “yes”, at what intervals?
   (i) hourly
   (ii) daily
   (iii) weekly
   (iv) monthly
   (v) at significant events in problem resolution
   (vi) other (please specify)

(f) Do you think there are any problems with the system that you use to manage your subdivision’s long term problems?

7. In an ideal world, how would you like to improve your problem management system?

8. Is there anything else that you think I should know about problem management?
Appendix F

Case-study client’s responses to Client Questionnaire
Responses

1. High-level goals:

What do you consider to be the main high-level goals in your domain of responsibility? For each goal can you supply the following information:

(a) A name and description for the goal.
(b) The name of the person or agency that assigned the goal to you, and the capacity in which they were acting.
(c) The kind of goal this is.
(d) The importance of this goal.

Please use copies of form Form-1 to supply this information.

Form-1: High-level goal:

(a) Goal name: "USER SUPPORT"

Goal description:

"To provide computer support to all the users. Provide assistance via the helpdesk, computer officers and helpsheets. Provide a means for users to report problems and for the problems to be investigated and, if possible corrected.

(b) Source of goal (indicate name and role):

"FCC" (i.e. Faculty Computing Committee)

(c) Goal type (tick up to two options):

- increase / o avoid
- maximise / o cease from
- achieve / o minimise
- maintain o decrease

(d) Importance of goal (tick one option):

/ o Highest importance \/
- Important
- Not very important
Form-1: High-level goal:

(a) Goal name: "COMPUTER SERVICE"

Goal description:

"To maintain the computer systems within our responsibility and ensure we meet the availability targets."

(b) Source of goal (indicate name and role):

"FCC"

(c) Goal type (tick up to two options):

- increase
- avoid
- maximise
- cease from
- achieve
- minimise
- maintain
- decrease

(d) Importance of goal (tick one option):

- Highest importance
- Important
- Not very important

Form-1: High-level goal:

(a) Goal name: TECHNICAL SUPPORT

Goal description:

"To offer technical support, in the form of advice, to the faculty in order that the faculty can make strategic decisions."

(b) Source of goal (indicate name and role):

"PROF. KEN JUKES / PCO’s BOSS" (i.e. the client’s manager)

(c) Goal type (tick up to two options):

- increase
- avoid
- maximise
- cease from
- achieve
- minimise
- maintain
- decrease

(d) Importance of goal (tick one option):
Form-1: High-level goal:

(a) Goal name: LIAISON

Goal description:

"To liaise with ITS and help define the strategy for computer services for the University."

(b) Source of goal (indicate name and role):

"Self-generated"

(c) Goal type (tick up to two options):

- increase
- minimise
- achieve
- maintain
- avoid
- cease from
- minimise
- decrease

(d) Importance of goal (tick one option):

- Highest importance
- Important
- Not very important

Form-1: High-level goal:

(a) Goal name: TECHNICAL INVESTIGATION

Goal description:

"To investigate new technologies and systems for possible implementation within the faculty"

(b) Source of goal (indicate name and role):

"Self-generated"

(c) Goal type (tick up to two options):

- increase
- minimise
- avoid
APPENDIX F. CASE-STUDY CLIENT’S RESPONSES TO CLIENT QUESTIONNAIRE

(o) maximise  (o) cease from
(o) achieve    (o) minimise
(o) maintain   (o) decrease

(d) Importance of goal (tick one option):

  o Highest importance  /
  o Important            \/
  o Not very important

Form-1: High-level goal:

(a) Goal name: TRAINING

  Goal description:

"To encourage the computer officers to develop their skills and knowledge."

(b) Source of goal (indicate name and role):

"Self-generated"

(c) Goal type (tick up to two options):

  /  
  o increase       o avoid
  o maximise      o cease from
  o achieve       o minimise
  o maintain      o decrease

(d) Importance of goal (tick one option):

  o Highest importance  /
  o Important            \/
  o Not very important

2. Stability of goals:

How frequently do your goals change (tick (a) or (b)):

  (a) Goals in my domain change frequently.  /
  (b) Goals in my domain are relatively stable.  \/

3. Domain structure:
Can you describe the high-level structure of your domain of responsibility. For each sub-division in your domain of responsibility can you supply the following information:

(a) The name of the sub-division.
(b) The sub-division leader’s name and their role name.

(c) Goals that you have assigned to the sub-division.
   For each goal, can you supply the following information:
   o Goal description
   o Goal type
   o Goal importance

(d) The nature of the sub-division’ work

Please use copies of form Form-2 to supply this information.

Form-2: Sub-division:

(a) Sub-division name:
"Help desk"

(b) Sub-division leader’s name and role name:
"LIZ DAVIES/HELP DESK MANAGER"

(c) Goals you have assigned to the sub-division:

Please use copies of form Form-3 to describe sub-division goals.

Form-3: Sub-division goal:

(a) Sub-division name:
"HELP DESK"

(b) Goal name:
"USER SUPPORT/PROBLEM LOGGING"

   Goal description:
   "To provide support to users in terms of offering help sheets,
manuals and minimal technical advice. Also provide a facility for users to log faults. Also provide a means of communicating the problems to the support teams and to monitor their progression to keep the users informed."

(c) Goal type (tick up to two options):

<table>
<thead>
<tr>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>increase</td>
</tr>
<tr>
<td>avoid</td>
</tr>
<tr>
<td>maximise</td>
</tr>
<tr>
<td>cease from</td>
</tr>
<tr>
<td>achieve</td>
</tr>
<tr>
<td>minimise</td>
</tr>
<tr>
<td>maintain</td>
</tr>
<tr>
<td>decrease</td>
</tr>
</tbody>
</table>

(d) Importance of goal (tick one option):

<table>
<thead>
<tr>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highest importance</td>
</tr>
<tr>
<td>Important</td>
</tr>
<tr>
<td>Not very important</td>
</tr>
</tbody>
</table>

Form-2: Sub-division:

(a) Sub-division name:

"UNIX SYSTEM SUPPORT"

(b) Sub-division leader's name and role name:

"PHILIP NAYLOR"

(c) Goals you have assigned to the sub-division:

Please use copies of form Form-3 to describe sub-division goals.

Form-2: Sub-division:

(a) Sub-division name:

"P.C. System Support"

(b) Sub-division leader's name and role name:

"SAM Hollyer"

(c) Goals you have assigned to the sub-division:

Please use copies of form Form-3 to describe sub-division goals.
goals.

Form-2: Sub-division:

(a) Sub-division name:

"Oracle Support"

(b) Sub-division leader's name and role name:

"Clare Williams"

(c) Goals you have assigned to the sub-division:

Please use copies of form Form-3 to describe sub-division goals.

Form-2: Sub-division:

(a) Sub-division name:

"Small Systems Support"

(b) Sub-division leader's name and role name:

"Alan Price"

(c) Goals you have assigned to the sub-division:

Please use copies of form Form-3 to describe sub-division goals.

Form-2: Sub-division:

(a) Sub-division name:

"Network Support"

(b) Sub-division leader's name and role name:

"Steve Allman"

(c) Goals you have assigned to the sub-division:
Please use copies of form Form-3 to describe sub-division goals.

4. Domain tasks:

Can you describe the main high-level tasks that you carry out in your domain of responsibility. For each task in you domain of responsibility can you supply the following information:

(a) A name and description for the task.

(b) The task’s importance.

Please use copies of form Form-9 to supply this information.

Form-9: High-level tasks:

(a) Task name:

"SUPPORT COMPUTER SYSTEMS"

Task description:

"To ensure that all computer systems are maintained in working order. That software is upgraded in order to improve the reliability of the systems and to assist in the support of the systems."

(b) Task importance:

//

o Very important
o Important
o Not very important

Form-9: High-level tasks:

(a) Task name:

"User Support"

Task description:

"Offer technical support to all users facing difficulties. This is achieved by providing help sheets and offering one to
one sessions to help with the problem."

(b) Task importance:

- Very important
- Important
- Not very important

Form-9: High-level tasks:

(a) Task name:

"PACKAGE SUPPORT"

Task description:

"Where possible offer technical support for some of the installed packages."

(b) Task importance:

- Very important
- Important
- Not very important

Form-9: High-level tasks:

(a) Task name:

"Print service"

Task description:

"Offer a service where users can print laser printouts which can then be collected from the help desk."

(b) Task importance:

- Very important
- Important
- Not very important

Form-9: High-level tasks:

(a) Task name:
"Problem reporting"

Task description:

"Allow problems to be reported to the helpdesk which can then be allocated to relevant system support team for remedy."

(b) Task importance:

- Very important
- Important
- Not very important

Form-9: High-level tasks:

(a) Task name:

"Technical Support"

Task description:

"Offer technical support to the faculty so that strategic decisions can be made."

(b) Task importance:

- Very important
- Important
- Not very important

Form-9: High-level tasks:

(a) Task name:

"Investigate new technology"

Task description:

"Actively look at new technology both hardware and software with a view of its use within the faculty."

(b) Task importance:

- Very important
- Important
Form-9: High-level tasks:

(a) Task name:
"installation"

Task description:
"Install new hardware/software systems within the faculty and configure for use."

(b) Task importance:

/ 

o Very important \/

o Important \

o Not very important

Form-9: High-level tasks:

(a) Task name:
"Lab support"

Task description:
"Maintain and support the labs to ensure that they are fully functional. Try and maintain an environment which is pleasant to work in."

(b) Task importance:

/ 

o Very important \/

o Important \

o Not very important

Form-9: High-level tasks:

(a) Task name:
"SELLING CONSUMABLES/MATERIALS"

Task description:
"Sell discs and manuals and some teaching materials."

(b) Task importance:

- Very important
- Important
- Not very important

5. Planned major changes:

Can you outline any major changes that you have planned for your domain of responsibility.

"Restructure teams into a User Support Oriented structure. Implement a problem logging and tracking system."

6. Common Goals:

Are there any goals which you have assigned in common to all of your sub-divisions.

Please use copies of form Form-4 to supply this information.

Form-4: Common goal:

(a) Goal name:

"SUPPORT COMPUTER SYSTEM"

Goal description:

"Ensure that all systems are adequately supported."

(b) Goal type (tick up to two options):

- increase
- avoid
- maximise
- cease from
- achieve
- minimise
- maintain
- decrease

(c) Importance of goal (tick one option):

- Highest importance
- Important
Form-4: Common goal:

(a) Goal name:
"INVESTIGATE NEW TECHNOLOGY"

Goal description:
To keep abreast of industry changes in the use of new technologies.

(b) Goal type (tick up to two options):
/  
  o increase /  o avoid 
  o maximise /  o cease from 
  o achieve /  o minimise 
  o maintain /  o decrease 

(c) Importance of goal (tick one option):
/  
  o Highest importance /  
  o Important /  
  o Not very important 

7. Problems:

What do you perceive to be the main problem areas within your domain of responsibility? For each problem area, can you supply the following information:

(a) A name and description for the problem.

(b) The severity of the problem.

(c) The frequency of occurrence of the problem.

(d) The goal(s) that the problem impacts the most.

Please use copies of form Form-5 provided to supply this information.
Form-5: Problems:

(a) Problem name:
"Problem logging"

Problem description:
"Logging and tracking where a problem is currently, after it has been communicated to the helpdesk."

(b) Problem severity (tick one option):

- Extremely severe
- Very severe
- Severe
- Not very severe
- Mild

(c) Frequency of occurrence of problem (tick one option):

- Continuous
- Weekly
- Other
- Hourly
- Monthly
- (please specify)
- Daily
- Yearly

(d) Impacted goals:

Please use copies of form Form-6 to supply this information.

Form-6: Impacted goals:

(a) Problem name:
"Problem logging"

(b) Goal name:
"User support"

Goal description:
Nature of impact:
Form-6: Impacted goals:

(a) Problem name:
"User support/problem logging"

(b) Goal name:
Goal description:
Nature of impact:

Form-5: Problems:

(a) Problem name:
"User focus"

Problem description:
"Not focusing on the customer as the main reason we are here."

(b) Problem severity (tick one option):
- Extremely severe
- Very severe
- Severe
- Not very severe
- Mild

(c) Frequency of occurrence of problem (tick one option):
- Continuous
- Weekly
- Monthly
- Daily
- Hourly
- Other
- Yearly

(d) Impacted goals:
Please use copies of form Form-6 to supply this information.

8. Improvements

What aspect of your domain of responsibility would you like to improve (apart, that is, from eliminating the problems you
have described above)? For each aspect could you supply the following information:

(a) The improvement that you would like (preferably quantified).

(b) The goals that the improvement would impact.

(c) The importance of the improvement.

Please use copies of form Form-7 to supply this information.

Form-7: Improvements:

(a) Improvement name:

"FASTER SERVICE"

Description of improvement:

"Improve the turnaround of support within the team. Again tied up with logging ans tracking of problems. Problems can slip and not get attention as they are not listed anywhere."

(b) Importance of this improvement:

- Extremely important
- Very important /
- Important /
- Not very important
- Mildly important

(c) Impacted goals:

Please use copies of form Form-8 to supply this information

Form-8: Impacted goals:

(a) Improvement name:

"Faster service"

(b) Goal name:

"User support/Problem logging"
Goal description:
Nature of impact:

Form-8: Impacted goals:

(a) Improvement name:
"Faster service"

(b) Goal name:
"User support"

Goal description:
Nature of impact:

Form-8: Impacted goals:

(a) Improvement name:
"Faster service"

(b) Goal name:
"Technical Investigation"

Goal description:
Nature of impact:

Form-7: Improvements:

(a) Improvement name:
"Liaise with Academics"

Description of improvement:
"Improve the relationship with teaching staff so that they know what we are doing and we know what they are doing."

(b) Importance of this improvement:
o Extremely important
o Very important /
o Important \/
o Not very important
o Mildly important

(c) Impacted goals:

Please use copies of form Form-8 to supply this information

Form-8: Impacted goals:

(a) Improvement name:
"Liaise with academics"

(b)
Goal name:
"User support/Problem logging"

Goal description:
Nature of impact:

Form-8: Impacted goals:

(a) Improvement name:
"Liaise with academics"

(b)
Goal name:
"User support"

Goal description:
Nature of impact:

Form-8: Impacted goals:

(a) Improvement name:
"Liaise with academics"

(b)
Goal name:

"Technical Investigation"

Goal description:
Nature of impact:
APPENDIX F. CASE-STUDY CLIENT’S RESPONSES TO CLIENT QUESTIONNAIRE

Improving the questionnaire

Do you think that the information you have supplied here would allow the questionnaire analyst to build up a comprehensive picture of both the context of your domain of responsibility and of its main problem areas. If not, can you indicate below what additional aspects you feel ought to be covered.

‘‘It was difficult for me to define the goals & task as they seem to overlap. It was also difficult to complete a list of goals due to the split. It was hard to think of what I had covered. I am sure I have missed goals.’’

10. Explanatory notes:

Which questions would benefit form having additional explanatory notes?

11. Time spent:

How long did it take you to complete this questionnaire?

‘‘2 hours’’

‘‘When I got to the end, I probably didn’t spend as much time on the problems & improvements. I found it difficult looking for improvements for a service department. I felt I need to talk to the users for their views really.’’
Appendix G

Validate client’s view of the served system

Section 6.3 of chapter 6, beginning on page 95, presented a process for validating a client’s view of the served system. That process was followed in the case study. The results that were obtained are presented below.

Step 1: Search for and resolve anomalies in the client’s questionnaire data

During the process of tabulating and modelling the data contained in the answers given by the client to the questionnaire questions, a number of anomalies were spotted. In order to resolve them, each was rephrased as a question for the client. The client was asked these questions. The questions and answers are given below (with the answers in bold).

CLARIFICATION: Questions requiring the client to clarify an issue.

1. Re the high-level goal User Support, in sentence 2 is it “computer officer” or “officers” or “office”?

   “officers”

2. With respect to the high-level goal Computer Service, would it be appropriate to also tick the “maintain” option for goal type?

   yes

3. At the moment it looks as though you have only assigned one goal to a subdivision, i.e. USER SUPPORT/PROBLEM LOGGING to the HELP DESK subdivision. Have you in fact also assigned goals to other subdivisions? For example, have you assigned goals in relation to your USER SUPPORT and COMPUTER SERVICE goals to the UNIX SYSTEM SUPPORT subdivision?

   The client had intended that the two “Common Goals” that were identified—SUPPORT COMPUTER SYSTEMS and INVESTIGATE NEW TECHNOLOGY—would be
generic goals. Each would be specialised for every identified domain subdivision apart from the HELP DESK. Each specialised goal would be assigned to its corresponding subdivision leader. For example, SUPPORT UNIX SYSTEM would be assigned to UNIX SYSTEM MANAGER and similarly for INVESTIGATE NEW UNIX TECHNOLOGY.

4. Re the goal assigned to the Help Desk Manager viz “User Support/ Problem Logging”, does the second part, i.e. “Also provide a facility for users to log faults, etc.”, express goals you would like to be able to assign to her, rather than goals that you have actually assigned to her?

No. The whole goal has been assigned and a procedure is in place to support it.

For each user problem, Liz can record on a pre-printed form details about the problem and its owner, viz “Name”, “User Name”, “Machine”, “Package”, “Error”, “Time”, “Date”, “Additional information”, “F.A.O.”, and “CSLOG Number”.

She can then either pass the problem to the Computing Officer who she deems most appropriate to deal with it, or deal with it herself.

Apparently this system is not used very often.

Appendix H (see page 283) presents a RAD model of, and questions derived through reflection upon, the process just described.

5. Is the common goal “Support computer systems” the same as the high-level goal “computer service”?

Yes. “Support computer systems” is intended to be a sub-goal of “Computer service”, each one specialised for a particular system - Unix, PC, etc.

6. Is the common goal “investigate new technology” the same as the high-level goal “technical investigation”?

See Item 5, on page 276, above.

7. Re the IMPROVEMENT “Faster Service”, does “team” refer to the set of computer officers?

Yes.

8. Apart from Liz Davies, have you asked other COs to respond to user’s problems?
Other COs, apart from Liz Davies, have been assigned the goal of responding to users’ problems. Currently these problems are communicated to COs either by Liz Davies or by a member of staff.

ELABORATION: Questions requiring the client to elaborate an issue.

1. Re the high-level goal Liaison, what does “ITS” stand for?

   ITS stands for Information Technology Services; it is the old name for the Computer Services Department.

2. Can you elaborate upon your plans to restructure the team into a user support oriented structure?

   The restructuring is intended to help meet the following goals.
   (a) To help move the emphasis of COTS work away from systems and more towards users
   (b) To allow system managers to focus on longer term issues, e.g. strategy formulation, rather than day-to-day issues like users’ problems. To free them from interrupts.

   The mechanism for achieving these goals is as follows. An Assistant CO in each sub-division would receive problems from Liz or from staff members. They would try to solve the problems. A CO would only be involved for unsolvable problems. The Assistant COs might be organised into a team—the User Support Team—under a User Support Manager. User Support Manager tasks might include the following: identify required helpsheets, manuals, and staff/student training. (R.E. NOTE: the possibilities for designed interaction between problem owners, the Help Desk Manager, Assistant Computer Officers, User Support Manager, and the Computer Officers are numerous and need to be explored.)

3. Will your new structure and new problem logging system be related? If so how?

   Yes; see Item 2, on page 277, above.

4. Re the PROBLEM “Logging and tracking where a problem is after it has been communicated to the help desk”, is the problem one of the following:
   a) logging problems
   b) tracking problems
   c) determining problem status as and when required
   d) (a), (b) and (c)
   e) (d) plus more. Can you elaborate?
APPENDIX G. VALIDATE CLIENT’S VIEW OF THE SERVED SYSTEM

The answer is (e), i.e. (a), (b), and (c) plus more. Additional required functionality includes the following:

(a) Keeping each user informed of the status of each of their problems
(b) Generating statistics on problems
(c) Recording solutions for errors and re-using the solutions
(d) Provision of a database of problem-solution pairs

(R.E. NOTE These requirements will need to be refined and justified in terms of acknowledged goals.)

5. Re IMPROVEMENT “Faster Service”, does “Improve turnaround for support, etc.” refer to support for problem solving? Can you elaborate this description? Yes. (R.E. NOTE: no answer was provided to the second question.)

ADDITIONAL INFORMATION Questions requiring additional information to be supplied by the client.

1. Re the sub-divisions, do their leaders have role names?

   The form of the name is:

   The name of the technology, e.g. Unix, followed by “System Manager”. For example Unix System manager

   But Network Controller, not Network System Manager.

2. Where does Julia Dawson fit into the picture?

   (R.E. NOTE: No answer for this question.)

3. Who decides what “help sheets ” will be made available?

   Helpsheets are produced by System Managers either proactively or under pressure from the users via the Help Desk Manager

4. Who produces the required “helpsheets”?

   System managers

5. Who decides what manuals will be made available?

   System Managers responding to staff requests.

CONFIRMATION: Questions requiring a confirmation (or otherwise) from the client.

1. Are the new high-level goal names that I have suggested acceptable to you?

   Yes.
APPENDIX G. VALIDATE CLIENT’S VIEW OF THE SERVED SYSTEM

2. Can you confirm the details of the “Decomposition of the “User Support” goal”?

(R.E. NOTE: no answer obtained.)

POSSIBLE INCOMPLETENESS: Questions where the client may have to supply more information.

1. For problems and improvements, you have not described the nature of their impact on affected goals. Can you say why you have not done so? Are you able to outline their impact now?

(R.E. NOTE: no answer obtained.)

NEXT STAGE INFORMATION: Questions to the client about issues associated with the next stage of the method.

1. Can you elaborate what you mean by “users”? For example, do you distinguish different kinds of users?

There are different kinds of users. For example there are staff users. These may be sub-divided into academic staff and administrative staff. There are also student users. This group may also be sub-divided.

2. What sort of “problems” do users report?

The kinds of problems reported by users include the following:

(a) Problems with a system, e.g. a problem with a Unix command argument not working as described.
(b) A problem that a user perceives he or she has due to lack of knowledge.
(c) A hardware problem.
(d) Software problem.
(e) An operating system problem.
(f) An application package problem.
(g) User account-related problems: e.g. password, disc space, or configuration file.
(h) Performance problem
(i) Operating System usage problem

3. In what ways is your department currently made aware of problems?

Users report problems to individual COs or to the Help Desk in the following ways:

(a) Personal contact
APPENDIX G. VALIDATE CLIENT’S VIEW OF THE SERVED SYSTEM

(b) Telephone
(c) Email
(d) Internal mail memo
(e) Fax

Also COs monitoring systems identify problems.

4. Are certain problem types associated with certain problems?

(R.E. NOTE: no information obtained.)

Step 2: Validate questionnaire data with the client:

The client confirmed that the tables and diagrams accurately reflected his view of his own domain.

Step 3: Identify key areas to focus subsequent investigation upon

(a) Identify the client’s most important high-level goals:

1. The client has identified “User Support” as the only goal of “Highest Importance”.

The description of the User Support goal is:

“To provide computer system support to all users. To provide assistance via the help desk, computer officers, and help sheets. Provide a means for users to report problems and for these problems to be investigated and if possible corrected.” (See table 7.1, on page 139.)

This goal seems to comprise two subgoals:

(i) Ongoing provision of assistance to users.
(ii) Managing users’ problems.

“User Support”, will remain a goal for the foreseeable future.

It seems as though the “Help Desk” “subdivision” is the only “subdivision” assigned responsibility for meeting this goal. In the context of this assignment, the client refers to the goal as “User Support/Problem Logging” and elaborates it slightly so that it becomes:

“To provide support to users in terms of offering help sheets, manuals and minimal technical advice. Also provide a facility to log faults. Also provide a means of communicating the problems to the support teams and to monitor their progression to keep users informed.” (See table 7.2, on page 141.)

As recorded earlier, a procedure is in place to support this goal: For each user problem, the Helpdesk System Manager can record
on a pre-printed form details about the problem and its owner, viz “Name”, “User Name”, “Machine”, “Package”, “Error”, “Time”, “Date”, “Additional information”, “F.A.O.”, and “CSLOG Number”. She can then either pass the problem to the Computing Officer who she deems most appropriate to deal with it, or deal with it herself. Apparently this system is not used very often. Appendix H, beginning on page 283, presents a RAD model of, and questions derived through reflection upon the process just described.

(b) Identify the client’s most important problems

1. The client has identified a “Problem Logging” “problem” as “severe” and “continuous”; the description of this “problem” is: “Logging and tracking where a problem is currently after it has been communicated to the help desk”. (See table 7.4, on page 142.)

This “problem” impacts the goals “User Support” and its variant “User Support/Problem Logging”, which are goals of the “Highest Importance” (see table 7.5, on page 143). In particular it impacts the “problem logging and tracking and solving” aspect of the goals.

c) Identify the clients most important improvements

1. The client has identified the “Improvement” “Faster Service”. This has the following description:

“Improve turnaround of support within the team. Again tied up with logging and tracking problems. Problems can slip and not get attention if they are not listed anywhere.” (See table 7.6, on page 144.)

It impacts the same aspect of the “User Support” and “User Support/Problem Logging” goals as the problem identified above (see table 7.7, on page 144).

The client intends to implement a problem logging and tracking system in the near future.

In the Elaboration part of section 7.3.1, item 2 (see page 277)provides more detail of the client’s plan to restructure the team into a user support oriented structure. This restructuring is intended to help meet the following goals:

• To help to move the emphasis of COTS work away from systems and more towards users;
• To allow system managers to focus on longer-term issues, e.g. strategy formulations, rather than day-to-day issues like users’ problems.

The first goal can be viewed as helping to reduce the User Focus problem (see table 7.4, page 142). The second goal was not
stated originally by the client. However it is also a goal of the subsystem managers as the sequel shows.

The client proposes the following mechanism for achieving these goals:

An Assistant CO in each sub-division would receive problems either from the Helpdesk manager or from staff members. They would try to solve the problems. A CO would only be involved for unsolvable problems. The Assistant COs might be organised into a team—the User Support Team—under a User Support Manager. User Support Manager tasks might include the following:

1. identify required helpsheets, manuals and staff/student training.

(d) Identify the intersection of (a) with (b) and/or (c).

**Step 4: Recommend to the client area(s) to focus subsequent investigation upon**

My recommendation to Jon (client) is that in the subsequent investigation we focus on the way his domain of responsibility handles users’ problems, with a view to achieving significant improvements. In particular we should take the following actions:

1. investigate how user problems are currently handled within Computer Services;
2. investigate users’ attitudes to the way their problems are currently handled;
3. start collecting anecdotal and quantitative data on user problems;
4. start to investigate how this problem, i.e. the management of user problems, is handled elsewhere.
Appendix H

Supporting the User
Support/Problem Logging

goal
Chapter 7 describes how the client in the case study completed a questionnaire, how this questionnaire was analysed, and how, acting as the RE, an attempt was made to resolve with the client the anomalies that were raised. This occurred in the validation stage, stage 1.1.3, of the process. There, Item 4 of Item G records a question that was put to the client about whether the second half of the goal User Support/Problem Logging had been assigned to the Help desk Manager Liz Davies.

The second half of the Support/Problem Logging goal is as follows:

“Also provide a facility for users to log faults. Also provide a means of communicating the problems to the support teams and to monitor their progression and to keep the users informed.”

The answer given was that the whole goal has been assigned to Liz Davies. The mechanism which currently attempts to satisfy the goal is as follows. For each user problem Liz can record on a pre-printed form details about the problem and its owner viz “Name”, “User name”, “Machine”, “Package”, “Error”, “Time” “Date”, etc. She can then either pass the problem to the CO that she considers most appropriate to deal with it, or deal with it herself.

This process is shown in figure H.1 below.

Analysis of the model raised the following questions:

1. What happens if the Help Desk Manager tries to solve the problem and fails? Does she then pass on the problem to the appropriate Computer Officer? Presumably, yes.

2. What happens if a Computer Officer fails to solve the problem? Does he or she involve other Computer Officers? What happens if they all fail?

3. If the Computer Officer solves the problem, does he or she communicate the solution to the Help Desk Manager, the user, both, or sometimes one and sometimes the other? In the latter case, what determines the target?

These questions helped to inform the Questionnaire that was administered to each Computing Officer.
Figure H.1: The resolve user-problem process
Appendix I

User questionnaire
Letter of introduction

The following e-mail containing the user questionnaire was sent to all members of the Faculty of Computer Studies and Mathematics:

Hi,

Jon Ward, the Principal Computing Officer, and I are undertaking a review of the services offered to CSM IT users by the Computer Officers (located in 3P12). The main purpose of the review is to determine the requirements for an enhanced "user-problem management system".

As part of this review we would like to find out how you, the users, feel about the quality of the services which are provided currently by the Computer Officers including the Help Desk facility. To this end we are asking you to complete the attached questionnaire (this should only take ten to fifteen minutes), and then to return it to me (Stewart Green) either by email or in hard copy form (either in the folder outside 2P32, or in my pigeon-hole) by Friday the 13th of December.

Thanks in advance for your help.

Stewart and Jon

COTS user questionnaire:
------------------------

1. To which of the following groups do you belong? (Please tick one group)

   Stage 1 student
   Stage 2 student
   Stage 3 student
   Postgraduate student
   Academic staff
   Administrative staff
   Other - Please specify:

2. On how many occasions within the last two years have you contacted either the Help Desk (in 3P12) or a Computer Officer for any reason? (Please tick one option)

   Never
   One to ten times
   Ten times or more
3. How happy would you say you were with the quality of the overall service you receive from the Help Desk and the Computer Officers?  
(Please tick one option)  

Very happy  
Happy  
Neutral  
Unhappy  
Very unhappy  

4. How happy would you say you were with the quality of each individual service that you have used?  
(Please circle all relevant choices)  

(a) Reporting problems directly to the Help Desk:  

Very happy  Happy  Neutral  Unhappy  Very unhappy  

(b) Reporting problems directly to Computer Officers:  

Very happy  Happy  Neutral  Unhappy  Very unhappy  

(c) Asking for advice at the Help Desk:  

Very happy  Happy  Neutral  Unhappy  Very unhappy  

(d) Asking Computer Officers for advice:  

Very happy  Happy  Neutral  Unhappy  Very unhappy  

(e) Asking the Help Desk for information:  

Very happy  Happy  Neutral  Unhappy  Very unhappy  

(f) Asking Computer Officers for information:  

Very happy  Happy  Neutral  Unhappy  Very unhappy  

(g) Borrowing manuals from the Help Desk  

Very happy  Happy  Neutral  Unhappy  Very unhappy  

(h) Buying materials from the Help Desk  

288
APPENDIX I. USER QUESTIONNAIRE

Very happy  Happy  Neutral  Unhappy  Very unhappy

(i) Collecting printed output from the Help Desk

Very happy  Happy  Neutral  Unhappy  Very unhappy

5. If you indicated that you were either "very unhappy" or "unhappy" with the provision of any of the services (a) to (f) above, can you list below the circumstances(s) that caused these feelings. (Continue on the other side of the page, if necessary.)

6. Do you have any suggestions for additional services that you would like to see provided by either the Help Desk or Computer Officers or both?

7. Do you have any suggestions for improvements which could be made to the ways in which the current services are provided?

8. Would you be willing to join a small team to help to determine the requirements for an enhanced "problem management service"?

If yes, please write your name and login id.

Name:
Login id.:
Appendix J

Analysis and review of users’ attitudes towards, and perceived problems with, COTS services
Analysis and review of users’ attitude towards, and perceived problems with, COTS services

Purpose of report:

The purpose of this report is to provide the client with information about the attitudes of staff and students (the users) in the CSM faculty towards the provision of services by the COTS team, their perceptions of problems with this provision, their suggestions for ways to improve this provision, and their ideas about new services that COTS might provide.

After the client has read the report, a meeting will be arranged with him in order to discuss its content and, based on the discussion, to decide upon the direction of subsequent work. In particular it will be necessary to prioritise problem areas selected for subsequent work, and to choose which improvements and new services will be given further consideration.

Introduction:

The client for this project has been Jon Ward, who was, until recently, the manager of the COTS group. This group provides support for staff and students from the faculty of Computer Studies and Mathematics who use the various computer systems - Unix, PC, MAC, etc. - installed in the faculty.

Inevitably problems occur when staff and students use the computer systems; such problems are often reported to personnel manning the COTS Helpdesk or directly to a Computer Officer; and, usually, an attempt is made to solve the problem.

The client felt that the way in which the COTS group managed such user problems was itself sometimes creating further problems. For example, he had a feeling that some user problems were being inadvertently overlooked and eventually forgotten. As a consequence, the client and the RE agreed at an early meeting (25/10/96) that the RE would review the way that the COTS team managed user problems, with a view to proposing improvements. It was expected that the improvements would include a set of requirements for a computer-based, socio-technical system which would help the COTS team to manage user problems.

However, the approach to requirements engineering which is being developed dictates that the context of a problem be investigated before the problem itself. And, for this project, two studies to investigate the problem context were carried out. In one study, the high and low-level goals associated with the COTS domain were investigated. The results have already been reviewed by the client, who validated the goal structures which had been uncovered.

In the other study, a questionnaire was used to discover four things: first, what the attitude of users was to the provision of services by the COTS group; second, what particular problems the users perceived with the provision of these services; third, suggestions made by the users on ways to improve existing services; and, fourth, users’ suggestions for new services.

Why was it considered necessary to determine this information? The users are key stakeholders for most COTS systems since they are the recipients of services provided by COTS. For this reason, it was important to
determine their attitude towards the provision of these services. If they were already happy with the existing provision, then, even if there were “real” problems associated with it, such problems might be viewed by the client as relatively unimportant, and not requiring an immediate solution.

For the same reason, it was important to determine what problems the users perceived to be associated with the COTS domain. If they were not happy with the existing provision of services—and the intention of the client was to improve that provision—then understanding the causes of their unhappiness might be an important factor in remedying it.

The same reason justifies the attempt to elicit their views on improving existing services and introducing new services. In addition, their views here might be particularly valuable because they have developed out of real experience of being on the receiving end of services provided by the COTS group.

**Attitude of users to services provided:**

In order to assess the attitude of the faculty’s users towards the services provided by the COTS team, a user questionnaire was sent to the members of the faculty (1,656). The responses were analysed and the results of the analysis are presented in tables J.1 and J.1.

The following observations relate to the results of the analysis. 90% of the 89 users who completed the questionnaire, had used COTS services - contacting either the Helpdesk or a Computer Officer - between one and ten times between November 1994 and November 1996. And 40% of these had used COTS services more than ten times within that period. In other words COTS services were used widely and frequently.

In general, most respondents seemed positive about the quality of the various services received. For example, approximately two thirds indicated that they were either “happy” (H) or “very happy” (VH) with the quality of the overall COTS service.

While respondents seemed positive about asking both Computer Officers (COs) and Helpdesk personnel for advice or information, after reporting problems they tended to be happier with the quality of service received from COs (75% vs. 25% respectively either “very happy” or “happy”). Of course, this is the kind of result one would expect given that visiting a Computer Officer generally means visiting someone with expertise in a given area, whereas interacting with Helpdesk staff generally means interacting with students who often lack deep knowledge in a wide range of areas.

Although most respondents were “neutral” towards borrowing manuals and buying materials from the Helpdesk (55% and 65% respectively), this may have been because few of them had actually used either of these services, perhaps being unaware of their existence. Very few respondents indicated that they were “very unhappy” or “unhappy” with either service (5% and 10% respectively).

Finally, although 60% were positive about collecting printed output, nearly one fifth were either “unhappy” or “very unhappy” with this service. Most of these dissatisfied respondents were stage two students.
## APPENDIX J. ANALYSIS AND REVIEW OF USERS’ ATTITUDES TOWARDS COTS

### How happy are you with the quality of the overall service received from the Helpdesk personnel and Computer Officers?

<table>
<thead>
<tr>
<th>Rating</th>
<th>Proportion</th>
</tr>
</thead>
<tbody>
<tr>
<td>VH</td>
<td>13.92%</td>
</tr>
<tr>
<td>H</td>
<td>53.16%</td>
</tr>
<tr>
<td>N</td>
<td>27.85%</td>
</tr>
<tr>
<td>U</td>
<td>3.80%</td>
</tr>
<tr>
<td>VU</td>
<td>1.27%</td>
</tr>
</tbody>
</table>

### How happy are you reporting problems directly to the Helpdesk?

<table>
<thead>
<tr>
<th>Rating</th>
<th>Proportion</th>
</tr>
</thead>
<tbody>
<tr>
<td>VH</td>
<td>18.42%</td>
</tr>
<tr>
<td>H</td>
<td>35.53%</td>
</tr>
<tr>
<td>N</td>
<td>38.16%</td>
</tr>
<tr>
<td>U</td>
<td>6.58%</td>
</tr>
<tr>
<td>VU</td>
<td>1.32%</td>
</tr>
</tbody>
</table>

### How happy are you reporting problems directly to the Computer Officers?

<table>
<thead>
<tr>
<th>Rating</th>
<th>Proportion</th>
</tr>
</thead>
<tbody>
<tr>
<td>VH</td>
<td>26.15%</td>
</tr>
<tr>
<td>H</td>
<td>46.15%</td>
</tr>
<tr>
<td>N</td>
<td>33.85%</td>
</tr>
<tr>
<td>U</td>
<td>9.23%</td>
</tr>
<tr>
<td>VU</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

### How happy are you asking for advice at the Helpdesk?

<table>
<thead>
<tr>
<th>Rating</th>
<th>Proportion</th>
</tr>
</thead>
<tbody>
<tr>
<td>VH</td>
<td>16.46%</td>
</tr>
<tr>
<td>H</td>
<td>43.04%</td>
</tr>
<tr>
<td>N</td>
<td>30.38%</td>
</tr>
<tr>
<td>U</td>
<td>8.86%</td>
</tr>
<tr>
<td>VU</td>
<td>1.27%</td>
</tr>
</tbody>
</table>

### How happy are you asking for advice from the Computer Officers?

<table>
<thead>
<tr>
<th>Rating</th>
<th>Proportion</th>
</tr>
</thead>
<tbody>
<tr>
<td>VH</td>
<td>22.08%</td>
</tr>
<tr>
<td>H</td>
<td>40.26%</td>
</tr>
<tr>
<td>N</td>
<td>31.17%</td>
</tr>
<tr>
<td>U</td>
<td>6.49%</td>
</tr>
<tr>
<td>VU</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

### How happy are you asking at the Helpdesk for information?

<table>
<thead>
<tr>
<th>Rating</th>
<th>Proportion</th>
</tr>
</thead>
<tbody>
<tr>
<td>VH</td>
<td>14.67%</td>
</tr>
</tbody>
</table>
APPENDIX J. ANALYSIS AND REVIEW OF USERS’ ATTITUDES TOWARDS COTS

<table>
<thead>
<tr>
<th>How happy are you asking the Computer Officers for information?</th>
<th>Proportion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rating</td>
<td></td>
</tr>
<tr>
<td>VH</td>
<td>17.11%</td>
</tr>
<tr>
<td>H</td>
<td>40.79%</td>
</tr>
<tr>
<td>N</td>
<td>34.21%</td>
</tr>
<tr>
<td>U</td>
<td>6.58%</td>
</tr>
<tr>
<td>VU</td>
<td>1.32%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>How happy are you borrowing manuals at the Helpdesk?</th>
<th>Proportion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rating</td>
<td></td>
</tr>
<tr>
<td>VH</td>
<td>16.13%</td>
</tr>
<tr>
<td>H</td>
<td>24.19%</td>
</tr>
<tr>
<td>N</td>
<td>54.84%</td>
</tr>
<tr>
<td>U</td>
<td>3.23%</td>
</tr>
<tr>
<td>VU</td>
<td>1.61%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>How happy are you buying materials from the Helpdesk?</th>
<th>Proportion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rating</td>
<td></td>
</tr>
<tr>
<td>VH</td>
<td>8.33%</td>
</tr>
<tr>
<td>H</td>
<td>16.67%</td>
</tr>
<tr>
<td>N</td>
<td>65.00%</td>
</tr>
<tr>
<td>U</td>
<td>5.00%</td>
</tr>
<tr>
<td>VU</td>
<td>5.00%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>How happy are you collecting printed output from the Helpdesk?</th>
<th>Proportion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rating</td>
<td></td>
</tr>
<tr>
<td>VH</td>
<td>18.57%</td>
</tr>
<tr>
<td>H</td>
<td>41.43%</td>
</tr>
<tr>
<td>N</td>
<td>22.86%</td>
</tr>
<tr>
<td>U</td>
<td>14.25%</td>
</tr>
<tr>
<td>VU</td>
<td>2.86%</td>
</tr>
</tbody>
</table>

Table J.2: User satisfaction with COTS services (continued)
If one were using these data as indicators of areas where improvements might be contemplated, then perhaps the quality of the service provided by Helpdesk personnel after users report problems and, possibly, the distribution of printout might make sensible candidates for areas to review.

**Problems currently experienced by users, along with suggested improvements and new services:**

The user questionnaire also invited staff and students to identify problems they had with the provision of services by COTS; to suggest ways of improving the existing services; and to propose new services. By examining the content of the responses made by the respondents in each of these three areas, the RE was able to create a set of meaningful categories in such a way that it was possible to place each response in at least one category. The set includes categories such as “problem management”, “service quality”, and “printouts”, for example.

For users’ “problems”, table J.3 ranks the categories in descending order. So, for example, the category “service quality” contained the most responses, the category “problem management” the next highest number, and so on. For “suggested improvements”, table J.4 ranks the categories similarly, and for “suggested new services” table J.5 does the same job.

Table J.6 shows which “problem”, “improvement” and “new services” responses have been identified as belonging to the “service quality” category. Tables J.7, J.8, and J.9 are similarly organised for responses in the categories of “problem management”, “communication” and “printouts”. (Responses in the category of “sales” and “opening times” have been omitted due their “distance” from solutions requiring computer-based systems support.)

As table J.3 indicates, the “service quality” category contained the most “problem” responses (15). The “problem management” category contained the second highest number of responses (9). While the “printouts” and “sales” categories contained 7 and 6 responses respectively, none of the eighteen remaining categories contained more than 4 responses.

These results seem to provide some support for focusing subsequent work on the “service quality” problems rather than the “problem management” problems. However, there are two reasons why it is considered that this would be the wrong focus. First, table J.6 indicates that most of the “problems” concern the alleged attitude of some Helpdesk personnel towards users. But, it is difficult to see how any computer-based system could help to improve this attitude. Second, only 15% (approximately) of the respondents provided comments that could could be categorised as “service quality” responses. It is considered that this figure is too low to indicate that users had a major problem in this area.

It is worth noting that tables J.6 through J.9 contain many suggestions for both improvements to existing services and for new services. For example, it seems to the writer that improvement number ten in table J.7, i.e.
## APPENDIX J. ANALYSIS AND REVIEW OF USERS’ ATTITUDES TOWARDS COTS

<table>
<thead>
<tr>
<th>Problem</th>
<th>Number of comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service quality</td>
<td>15</td>
</tr>
<tr>
<td>Problem management</td>
<td>9</td>
</tr>
<tr>
<td>Printouts</td>
<td>7</td>
</tr>
<tr>
<td>Sales</td>
<td>6</td>
</tr>
<tr>
<td>Opening-closing times</td>
<td>4</td>
</tr>
<tr>
<td>Communication</td>
<td>3</td>
</tr>
<tr>
<td>Manuals</td>
<td>3</td>
</tr>
<tr>
<td>Systems availability</td>
<td>3</td>
</tr>
<tr>
<td>COTS - ITS relationship</td>
<td>2</td>
</tr>
<tr>
<td>Packages/technology</td>
<td>2</td>
</tr>
<tr>
<td>Self-preservation</td>
<td>1</td>
</tr>
<tr>
<td>COTS - STAFF relationship</td>
<td>1</td>
</tr>
<tr>
<td>Autonomous policy makers</td>
<td>1</td>
</tr>
<tr>
<td>System change management</td>
<td>1</td>
</tr>
<tr>
<td>Computer officer availability</td>
<td>1</td>
</tr>
<tr>
<td>Computer officer work pressure</td>
<td>1</td>
</tr>
<tr>
<td>Facilities</td>
<td>1</td>
</tr>
<tr>
<td>COTS - STUDENTS relationship</td>
<td>0</td>
</tr>
<tr>
<td>Helpdesk availability</td>
<td>0</td>
</tr>
<tr>
<td>Staff morale</td>
<td>0</td>
</tr>
<tr>
<td>Helpdesk work pressure</td>
<td>0</td>
</tr>
<tr>
<td>Systems support</td>
<td>0</td>
</tr>
</tbody>
</table>

Table J.3: Problems perceived by users (ordered by number of comments)
<table>
<thead>
<tr>
<th>Improvement</th>
<th>Number of comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>13</td>
</tr>
<tr>
<td>Problem management</td>
<td>11</td>
</tr>
<tr>
<td>Opening-closing times</td>
<td>8</td>
</tr>
<tr>
<td>Facilities</td>
<td>4</td>
</tr>
<tr>
<td>Service quality</td>
<td>4</td>
</tr>
<tr>
<td>Printouts</td>
<td>4</td>
</tr>
<tr>
<td>COTS - STAFF relationship</td>
<td>3</td>
</tr>
<tr>
<td>COTS - STUDENTS relationship</td>
<td>2</td>
</tr>
<tr>
<td>COTS - ITS relationship</td>
<td>1</td>
</tr>
<tr>
<td>Helpdesk availability</td>
<td>1</td>
</tr>
<tr>
<td>Staff morale</td>
<td>1</td>
</tr>
<tr>
<td>Sales</td>
<td>1</td>
</tr>
<tr>
<td>Autonomous policy makers</td>
<td>0</td>
</tr>
<tr>
<td>System change management</td>
<td>0</td>
</tr>
<tr>
<td>Manuals</td>
<td>0</td>
</tr>
<tr>
<td>Systems availability</td>
<td>0</td>
</tr>
<tr>
<td>Packages/technology</td>
<td>0</td>
</tr>
<tr>
<td>Self-preservation</td>
<td>0</td>
</tr>
<tr>
<td>Computer officer availability</td>
<td>0</td>
</tr>
<tr>
<td>Computer officer work pressure</td>
<td>0</td>
</tr>
<tr>
<td>Helpdesk work pressure</td>
<td>0</td>
</tr>
<tr>
<td>Systems support</td>
<td>0</td>
</tr>
</tbody>
</table>

Table J.4: Improvements to existing services suggested by users)


### Table J.5: New services suggested by users

<table>
<thead>
<tr>
<th>Service</th>
<th>Number of comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>6</td>
</tr>
<tr>
<td>Problem management</td>
<td>5</td>
</tr>
<tr>
<td>Facilities</td>
<td>5</td>
</tr>
<tr>
<td>Service quality</td>
<td>4</td>
</tr>
<tr>
<td>Packages/technology</td>
<td>4</td>
</tr>
<tr>
<td>Opening-closing times</td>
<td>3</td>
</tr>
<tr>
<td>Printouts</td>
<td>2</td>
</tr>
<tr>
<td>COTS - STAFF relationship</td>
<td>2</td>
</tr>
<tr>
<td>Sales</td>
<td>2</td>
</tr>
<tr>
<td>Systems availability</td>
<td>2</td>
</tr>
<tr>
<td>Computer officer availability</td>
<td>1</td>
</tr>
<tr>
<td>Systems support</td>
<td>1</td>
</tr>
<tr>
<td>COTS - STUDENTS relationship</td>
<td>0</td>
</tr>
<tr>
<td>COTS - ITS relationship</td>
<td>0</td>
</tr>
<tr>
<td>Helpdesk availability</td>
<td>0</td>
</tr>
<tr>
<td>Staff morale</td>
<td>0</td>
</tr>
<tr>
<td>Autonomous policy makers</td>
<td>0</td>
</tr>
<tr>
<td>System change management</td>
<td>0</td>
</tr>
<tr>
<td>Manuals</td>
<td>0</td>
</tr>
<tr>
<td>Self-preservation</td>
<td>0</td>
</tr>
<tr>
<td>Computer officer work pressure</td>
<td>0</td>
</tr>
<tr>
<td>Helpdesk work pressure</td>
<td>0</td>
</tr>
</tbody>
</table>
### Problems

1. Explanations too technical for some users.
2. Helpdesk staff patronising.
3. Poor service when dealing with uncommon machine configurations and/or software.
4. Unhelpful Helpdesk staff.
5. Unsympathetic Helpdesk staff.
6. Helpdesk staff sometimes cannot understand users’ problems.
7. Helpdesk staff unwilling to listen.
8. Helpdesk staff member abrupt and rude.
9. Helpdesk staff too busy.
10. Staff on Helpdesk know little about computers or systems supported.
11. Helpdesk staff unwilling to tackle problems they don’t understand.

### Improvements

1. Demonstrate that what has been provided does actually work.
2. Provide technical training for Helpdesk staff.
3. Provide customer relations training for Helpdesk staff.
4. Do not use freshers on the Helpdesk.
5. Do not use on the Helpdesk people who are unaware of the facilities which are supported in the faculty.
6. Complete all requests, e.g. access to student photos.
7. Hire more staff.
8. Ensure that at least one Unix person and one PC person is available whenever laboratories are scheduled.
9. Increase the number of staff covering each supported system.
10. Provide a 24 hour service.
11. Provide cover before 10 a.m.
12. Provide CO level cover during lunch hours.
13. Ensure that COs are around out of hours.

### New services

No new services were suggested in this area.

Table J.6: User responses in the “service quality” category
APPENDIX J. ANALYSIS AND REVIEW OF USERS’ ATTITUDES TOWARDS COTS

### Problems

1. Some problems reported by users are ignored and/or forgotten.
2. Managers’ problems are not always given a high enough priority.
3. COTS staff are too busy to help.
4. Helpdesk staff have insufficient knowledge to help.

### Improvements

1. Employ more computer officers.
2. Employ more Helpdesk personnel.
3. Enable computer officers to spend more time on problems.
4. Provide more technical training for Helpdesk personnel.
5. Provide customer handling training for Helpdesk personnel.
6. Do not use freshers on the Helpdesk.
7. Give priority to academic staff problems.
8. Increase staff coverage for each supported system, e.g. more staff support for MACs.
9. Provide Unix and PC support whenever the laboratories are open.
10. Allow users to e-mail problems to Helpdesk personnel and computer officers, i.e. publish relevant email points.

### New services

1. Identify a system expert for each system - someone who is prepared to field quite tough problems.
2. Have set times when computer officers would be dedicated to tackling user problems. Publicize them.
3. Publicize skill profiles for each member of the COTS team.
4. Log problems to prevent verbal requests from being lost.

Table J.7: User responses in the “problem management” category
“Provide Unix and PC support whenever the laboratories are open” is a sensible one worthy of further consideration.

Conclusion

The results of the user questionnaire seem to indicate that users are reasonably happy with the current provision of services by the COTS group and do not experience any major problems in common. Of the problems they have, the one that the writer considers is most likely to lead an improvement in quality of service provision through support by a computer-based system is the problem of recording and tracking problems reported by users. In this context it is worth noting that one “problem” response was: “Some problems reported by users are ignored and/or forgotten”; and that one suggestion for new services was “log problems to prevent verbal requests from being lost.”
### Problems

1. Sometimes system changes are made but the staff affected by the changes are not informed.

2. Some users are not aware of what services COTS do provide, e.g. some are unaware of the lending manuals service.

3. Some users are unaware of the details of some services, e.g. of what manuals are available to borrow.

### Improvements

1. Better liaison with academics.

2. Inform non-PC users when new PC software or approaches introduced.

3. Provide a catalogue with a one paragraph description of each piece of software.

4. Publish more information on what COTS can do for its users.

5. Present the same information in different ways to cater for different groups of users.

6. Publish more information and guidance on dial-in services and support for teleworking.

7. rewrite current information sheets to be more useful for tackling assignments.

8. Put all Helpdesk handouts online.

9. Staff should be informed by COTS personnel when a software licence is up for renewal.

### New services

1. Provide a basic guide to all students outlining what facilities and resources are available.

2. Display prominently a list of leaflets and booklets available from the Helpdesk.

3. Maintain an online bulletin covering e.g. system status, current problems, timescales, plans, etc.

4. “Problem management” points 1 to 3.

5. Publish project plans and timescales and whenever something is not going to be provided.

6. At the start of the first year provide a handout about what computers are available, what rooms they are in, what software they have and how to print from Unix.

Table J.8: User responses in the “communication” category
APPENDIX J. ANALYSIS AND REVIEW OF USERS’ ATTITUDES TOWARDS COTS

<table>
<thead>
<tr>
<th>Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Printouts misfiled.</td>
</tr>
<tr>
<td>2. Printouts filed long after being printed, e.g. two weeks.</td>
</tr>
<tr>
<td>3. The system for printing and distributing printouts is over-complicated.</td>
</tr>
<tr>
<td>4. Printouts are not free for students; 5p per sheet is too expensive.</td>
</tr>
<tr>
<td>5. Collecting printouts can be a problem near large assignment hand-in dates.</td>
</tr>
<tr>
<td>6. Quality of printouts often poor.</td>
</tr>
<tr>
<td>7. Need to queue sometimes for printouts.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Improvements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Is it not possible to print to laser printer in e.g. 2P21?</td>
</tr>
<tr>
<td>2. Have more printers to reduce or avoid queueing.</td>
</tr>
<tr>
<td>3. Reduce the cost of laser printout.</td>
</tr>
<tr>
<td>4. Monitor printers with more care and more frequently.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>New services</th>
</tr>
</thead>
<tbody>
<tr>
<td>No new services were suggested in this area.</td>
</tr>
</tbody>
</table>

Table J.9: User responses in the “printouts” category
Appendix K

Client’s feedback on the analysis and review of users’ attitudes towards COTS services

Client’s response to the results

The summary of questionnaire results and its analysis, discussed above, were written up as a report and presented to the client. His response to the report was obtained in an interview with him conducted by the RE. The RE posed a number of questions and the client’s answers were noted. After the interview the clients answers in note form were written up by the RE. The questions that were put to the client and his answers are presented below.

Do you have any general comments on the report?

Overall the client thought that it was a good report. He was particularly interested in the problems that users had and in their suggestions for improvements to existing services and for new services. This is because he felt that it is often difficult to understand what is going on when one is “stuck in the middle”. I.e. when one’s energies are taken up with managing the delivery of a service, it’s often difficult to stand back to get a clear idea of what users’ concerns really are.

He felt that some of the users’ suggestions would be difficult to achieve. For example, it would be hard to increase the number of Computing Officers due to financial constraints. On the other hand the client considered that some of the suggestions were very good and would be relatively easy to achieve.
APPENDIX K. CLIENT’S FEEDBACK ON THE ANALYSIS OF
USERS’ ATTITUDES TO COTS

Can you prioritise the problems (reported in the report) in the order that you would like me to investigate them?

In first position the client placed the lack of a good problem recording system.

The client saw communication problems as an almost equal priority with problem recording.

A number of other problems were discussed, e.g. printing, service quality and opening and closing times. However, it was clear from the client’s comments he had thought about each of these problems, and could put forward reasons for not treating them as problems to be tackled by the RE.

For example, for printing the client pointed out that the problems of charging and the need to print in 3P12 and come there to collect printouts were not in his control. The Dean, supported by the Faculty Computing Committee, had asked for charging, and faculty requirements take precedence over user requirements. The need to charge has led inevitably to the placement of laser printers in 3P12, a secure area, and the need to come to 3P12 to pay for and collect printouts. Although other ways of solving this problem had been considered, these technical solutions were judged too unwieldy.

For the service quality problems, the client indicated that financial constraints prevented him from employing another full-time professional Computer Officer to work on the Helpdesk. However, he was able to afford ten student assistants each year, and, because having one or more people working alongside the Helpdesk Manager on a shift relieves the pressure on that manager, this is thought to be a cost-effective measure.

With regard to opening and closing times, a mixture of financial constraints, contractual restrictions on hours worked, and the need to carry out support tasks such as backing up systems and starting up systems have led the client to consider the ten o’clock start as necessary and the inability to provide a 24 hour service as inevitable.

Did you find it useful to receive this list of user problems?

The client indicated that he had found it very useful to be made aware of user problems, for three main reasons. First, it was useful for him to see that some of the problems that he thought that users were experiencing were indeed being experienced by the users. Second, it was useful for him to be made aware of problems of which he was previously unaware. And third, the client thought that he could use the results in the independent report to make a case to higher management for more resources.

Did you find it useful to receive the suggestions for improvements to existing services?

Again the client indicated that he had found the list of improvement useful, for three main reasons. First, as before, improvement suggestions could be
APPENDIX K. CLIENT’S FEEDBACK ON THE ANALYSIS OF USERS’ ATTITUDES TO COTS

taken to management as part of a case made for additional services. Second, some of the suggestions could be given further consideration by COTS almost immediately. For example, the suggestion to “demonstrate that what has been provided does actually work” could be acted upon straight away. Third, knowing what people would like, even if it cannot be provided, is useful, because it allows COTS to explain to all users why they cannot provide a particular service. The idea here is that an appropriately informed user is likely to be more understanding and tolerant than an ill-informed one of COTS’ current level of provision of services. For example, COTS could publicise the fact that “ensuring COs are around out of normal hours” cannot be met because of contractual constraints.

Did you find it useful to receive suggestions for new services?

Again, the client indicated that he had found it useful to receive this list. The client would like COTS to be much more user-driven than at present in this area, so it was useful for him to read about new services requested by users that COTS staff had not already thought about. If this were to be the norm, it would enable COTS staff to start planning early to provide such new services.
Appendix L

The wider served system

Attitude of faculty users to COTS services provided

VH = Very Happy
H = Happy
N = Neutral
U = Unhappy
VU = Very Unhappy

Table A: Attitude of faculty users to COTS services provided

<table>
<thead>
<tr>
<th>Experience with COTS services?</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Never used:</td>
<td>4.49%</td>
</tr>
<tr>
<td>Used 1 - 10 times:</td>
<td>53.93%</td>
</tr>
<tr>
<td>used more than 10 times:</td>
<td>41.57%</td>
</tr>
</tbody>
</table>

How happy with the quality of overall service received from Helpdesk personnel and Computer Officers?

| VH   | 13.92% |
| H    | 53.16% |
| N    | 27.85% |
| U    | 3.80%  |
| VU   | 1.27%  |

How happy reporting problems directly to the Helpdesk?

| VH   | 18.42% |
| H    | 35.53% |
| N    | 38.16% |
| U    | 6.58%  |
| VU   | 1.32%  |

How happy reporting problems directly to the Computer Officers?
APPENDIX L. THE WIDER SERVED SYSTEM

VH 26.15%
H 46.15%
N 33.85%
U 9.23%
VU 0.00%

How happy asking for advice at the Helpdesk?

VH 16.46%
H 43.04%
N 30.38%
U 8.86%
VU 1.27%

How happy asking for advice from the Computer Officers?

VH 22.08%
H 40.26%
N 31.17%
U 6.49%
VU 0.00%

How happy asking at the Helpdesk for information?

VH 14.67%
H 37.33%
N 41.33%
U 6.67%
VU 0.00%

How happy asking the Computer Officers for information?

VH 17.11%
H 40.79%
N 34.21%
U 6.58%
VU 1.32%

How happy borrowing manuals at the Helpdesk?

VH 16.13%
H 24.19%
N 54.84%
U 3.23%
VU 1.61%

How happy buying materials from the Helpdesk?

VH 8.33%
How happy collecting printed output from the Helpdesk?

VH 18.57%
H 41.43%
N 22.86%
U 14.25%
VU 2.86%

Problems perceived by users

Table B: Problems perceived by users (ordered by number of comments):

<table>
<thead>
<tr>
<th>Issue</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service quality</td>
<td>15</td>
</tr>
<tr>
<td>Problem management</td>
<td>9</td>
</tr>
<tr>
<td>Printouts</td>
<td>7</td>
</tr>
<tr>
<td>Sales</td>
<td>6</td>
</tr>
<tr>
<td>Opening-closing times</td>
<td>4</td>
</tr>
<tr>
<td>Communication</td>
<td>3</td>
</tr>
<tr>
<td>Manuals</td>
<td>3</td>
</tr>
<tr>
<td>Systems availability</td>
<td>3</td>
</tr>
<tr>
<td>COTS - ITS relationship</td>
<td>2</td>
</tr>
<tr>
<td>Packages/technology</td>
<td>2</td>
</tr>
<tr>
<td>Self-preservation</td>
<td>1</td>
</tr>
<tr>
<td>COTS - STAFF relationship</td>
<td>1</td>
</tr>
<tr>
<td>Autonomous policy makers</td>
<td>1</td>
</tr>
<tr>
<td>System change management</td>
<td>1</td>
</tr>
<tr>
<td>Computer officer availability</td>
<td>1</td>
</tr>
<tr>
<td>Computer officer work pressure</td>
<td>1</td>
</tr>
<tr>
<td>Facilities</td>
<td>1</td>
</tr>
<tr>
<td>COTS - STUDENTS relationship</td>
<td>0</td>
</tr>
<tr>
<td>Helpdesk availability</td>
<td>0</td>
</tr>
<tr>
<td>Staff morale</td>
<td>0</td>
</tr>
<tr>
<td>Helpdesk work pressure</td>
<td>0</td>
</tr>
<tr>
<td>Systems support</td>
<td>0</td>
</tr>
</tbody>
</table>

Improvements to existing services suggested by users:

Table C: Improvements to existing services suggested by users:

<table>
<thead>
<tr>
<th>Issue</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>13</td>
</tr>
</tbody>
</table>
Problem management 11
Opening-closing times 8
Facilities 4
Service quality 4
Printouts 4
COTS - STAFF relationship 3
COTS - STUDENTS relationship 2
COTS - ITS relationship 1
Helpdesk availability 1
Staff morale 1
Sales 1
Autonomous policy makers 0
System change management 0
Manuals 0
Systems availability 0
Packages/technology 0
Self-preservation 0
Computer officer availability 0
Computer officer work pressure 0
Helpdesk work pressure 0
Systems support 0

New services suggested by users:

Table D: New services suggested by users:

-----------------------------------------
Communication 6
Problem management 5
Facilities 5
Service quality 4
Packages/technology 4
Opening-closing times 3
Printouts 2
COTS - STAFF relationship 2
Sales 2
Systems availability 2
Computer officer availability 1
Systems support 1
COTS - STUDENTS relationship 0
COTS - ITS relationship 0
Helpdesk availability 0
Staff morale 0
Autonomous policy makers 0
System change management 0
Manuals 0
Self-preservation 0
Computer officer work pressure 0

310
Helpdesk work pressure 0

User responses in the “service quality” category:

Table E: User responses in the “service quality” category:

PROBLEMS

1. Explanations too technical for some users.
2. Helpdesk staff patronising.
3. Poor service when dealing with uncommon machine configurations and/or software.
4. Unhelpful Helpdesk staff.
5. Unsympathetic Helpdesk staff.
6. Helpdesk staff sometimes cannot understand users’ problems.
7. Helpdesk staff unwilling to listen.
8. Helpdesk staff member abrupt and rude.
9. Helpdesk staff too busy.
10. Staff on Helpdesk know little about computers or systems supported.
11. Helpdesk staff unwilling to tackle problems they don’t understand.

IMPROVEMENTS

1. Demonstrate that what has been provided does actually work.
2. Provide technical training for Helpdesk staff.
3. Provide customer relations training for Helpdesk staff.
4. Do not use freshers on the Helpdesk.
5. Do not use on the Helpdesk people who are unaware of the facilities which are supported in the faculty.
6. Complete all requests, e.g. access to student photos.
7. Hire more staff.
8. Ensure that at least one Unix person and one PC person is available whenever laboratories are scheduled.
9. Increase the number of staff covering each supported system.
10. Provide a 24 hour service.
11. Provide cover before 10 a.m.
12. Provide CO level cover during lunch hours.
13. Ensure that COs are around out of hours.

NEW SERVICES

No new services were suggested in this area.
User responses in the “problem management” category:

1. Some problems reported by users are ignored and/or forgotten.
2. Managers’ problems are not always given a high enough priority.
3. COTS staff are too busy to help.
4. Helpdesk staff have insufficient knowledge to help.

User responses in the “communication” category:

1. Sometimes system changes are made but the staff affected by the changes are not informed.
2. Some users are not aware of what services COTS do provide, e.g. some are
unaware of the lending manuals service.

3. Some users are unaware of the details of some services, e.g. of what manuals are available to borrow.

IMPROVEMENTS

1. Better liaison with academics.
2. Inform non-PC users when new PC software or approaches introduced.
3. Provide a catalogue with a one paragraph description of each piece of software.
4. Publish more information on what CUTS can do for its users.
5. Present the same information in different ways to cater for different groups of users.
6. Publish more information and guidance on dial-in services and support for teleworking.
7. Rewrite current information sheets to be more useful for tackling assignments.
8. Put all Helpdesk handouts online.
9. Staff should be informed by CUTS personnel when a software licence is up for renewal.

NEW SERVICES

1. Provide a basic guide to all students outlining what facilities and resources are available.
2. Display prominently a list of leaflets and booklets available from the Helpdesk.
3. Maintain an online bulletin covering e.g. system status, current problems, timescales, plans, etc.
4. ‘Problem management’ points 1 to 3.
5. Publish project plans and timescales and whenever something is not going to be provided.
6. At the start of the first year provide a handout about what computers are available, what rooms they are in, what software they have and how to print from Unix.

User responses in the “Printouts” category:

Table H: User responses in the ‘‘Printouts’’ category:

PROBLEMS

1. Printouts misfiled.
2. Printouts filed long after being printed, e.g. two weeks.
3. The system for printing and distributing printouts is overcomplicated.
4. Printouts are not free for students; 5p per sheet is too expensive.
5. Collecting printouts can be a problem near large assignment hand-in dates.
6. Quality of printouts often poor.
7. Need to queue sometimes for printouts.

IMPROVEMENTS

1. Is it not possible to print to laser printer in e.g. 2P21?
2. Have more printers to reduce or avoid queueing.
3. Reduce the cost of laser printout.
4. Monitor printers with more care and more frequently.

NEW SERVICES

No new services were suggested in this area.
Appendix M

Subsystem owners’ responses to questionnaire
Introduction

This appendix presents the questionnaire responses produced by the Helpdesk manager and the other subsystem managers. Each person was interviewed separately in a session lasting from 30 minutes to one hour.

Questionnaire for the Helpdesk Manager

1. High-level goals:

As the Helpdesk Manager what do you consider are the main high-level goals of your domain of responsibility, the Helpdesk? For each goal can you supply the following information:

(a) A name and description for the goal.

(b) The name of the person or agency that assigned the goal to you, and the capacity in which they were acting. (This may include self-assigned goals)

(c) The kind of goal this is. (Increase, Maximize, Achieve, Maintain, Avoid, Cease from, Minimize, Decrease)

(d) The importance of this goal. (Highest Importance, Important, Not Highest Importance.)

(a) Satisfy users
   (a) Satisfy staff and users.
   (b) Self-assigned
   (c) Maintain
   (d) Highest-Importance

(b) Maintain a library
   (a) Provision of a library for staff and student use that would hold books and software manuals that are otherwise difficult to obtain.
   (b) Self-assigned
   (c) Inherited
   (d) Highest-Importance

(c) Eliminate the price of student printouts
   (a) Minimise or eliminate the cost to students of laser printout.
   (b) Self-assigned (approved by client)
   (c) Minimize or Avoid
   (d) The importance of this goal.

(d) Develop expertise in solving user problems
   (a) Build up expertise in solving user problems.
   (b) Self-assigned
   (c) Maximise
   (d) Highest-Importance
(e) Maintain positive attitude
   (a) Maintain a positive attitude towards helping users.
   (b) Self-assigned
   (c) No data
   (d) No data

(f) Help foreign students
   (a) Help foreign students.
   (b) Self-assigned
   (c) No data
   (d) No data

(g) Dispense advice to users on solving problems.
   (a) Advise on best way to solve problems.
   (b) No data
   (c) No data
   (d) No data

(h) Identify sources of help
   (a) Quickly identify relevant sources of help for users.
   (b) Self-generated
   (c) No data
   (d) No data

(i) Respond to e-mails within one day
   (a) Try to respond within one day to students’ e-mailed problems.
   (b) Self-generated
   (c) Maintain
   (d) Important

(j) Present positive image of Helpdesk group
   (a) Try to ensure that the Helpdesk group are seen to be a cohesive, helping group.
   (b) No data
   (c) No data
   (d) No data

2. Stability of goals:

   How frequently do your goals change (tick one of the following):

   (b) Goals in my domain of responsibility are relatively stable.

3. Do you consider that resolving staff problems, requests for advice, requests for information, etc. is one of the goals of the Helpdesk? (Please tick one option)

   (i) Yes
4. If you answered “yes” to the previous question, could you indicate how important the goal is to the Helpdesk. (Please tick one option)

   (iii) Highest importance

5. (a) What percentage of user problems, advice or information requests, etc. do you resolve on the spot while the user is with you? (Write down a percentage from 0% to 100%)

   90%

   (b) What percentage of user problems, advice or information requests, etc. do you pass on to another person or group? (Write down a percentage from 0% to 100%)

   10%

   (The Helpdesk Manager either writes down the problem and later passes it to a CO, or takes the user to see a CO, or tells the user to see a CO, or goes herself to explain the problem to a CO.)

   (c) What percentage of user problems, advice or information requests, etc. do you add to your own list of problems to be investigated and resolved? (Write down a percentage from 0% to 100%)

   0%

6. What are the main problems that you experience in managing the help desk? In particular what are the main problems that you experience in managing user problem management?

   (a) Rudeness from staff and students

   (b) Male chauvinism: users not wanting to have their problems managed by the female Helpdesk Manager

7. In an ideal world, how would you like to improve the Helpdesk problem management system?

   (a) The Helpdesk Manager should manage only the helpdesk. In particular he or she should not manage the budget for consumables like toner.

   (b) Another similar, adaptable, flexible, non-computer expert should be employed to work alongside the Helpdesk Manager and share the load.

   (c) A computer expert should be continuously available at the helpdesk to help to deal with user problems.

8. Is there anything else that you think I should know about problem management?
The management of the consumables budget detracts from the provision of a good helpdesk service.

Questionnaire responses for the Unix System Manager

1. High-level goals:

As the subdivision System Manager what do you consider are the main high-level goals of your domain of responsibility, the Helpdesk? For each goal can you supply the following information:

(a) A name and description for the goal.
(b) The name of the person or agency that assigned the goal to you, and the capacity in which they were acting. (This may include self-assigned goals)
(c) The kind of goal this is. (Increase, Maximize, Achieve, Maintain, Avoid, Cease from, Minimize, Decrease)
(d) The importance of this goal. Highest Importance, Important, Not Highest Importance

(a) Keep Unix systems running smoothly
   (a) Keep Unix systems running smoothly
   (b) Job description
   (c) Maintain
   (d) Highest-Importance

(b) User support.
   (a) Provide user support (troubleshooting).
   (b) Job description
   (c) Reduce or Avoid
   (d) Highest Importance (prefer Not Highest Importance)

(c) User education
   (a) Provision of information/education on Unix matters.
   (b) Self-assigned
   (c) Maintain
   (d) Not Highest Importance (prefer Highest Importance)

(d) Forecast needs
   (a) To forecast and meet future Unix-related needs
   (b) Job description.
   (c) Maintain
   (d) Important

(e) Maximise usability
   (a) Maximise usability of the Unix system
   (b) Self-assigned
(c) Maximise
(d) Important
(f) Justify Unix
   (a) Justify presence of the Unix system
   (b) Self-assigned
   (c) Maintain
   (d) Important

2. Stability of goals:

How frequently do your goals change (tick one of the following):

(b) Goals in my domain of responsibility are relatively stable.
   (However, the relative importance of goals changes between term-time and vacations. For example, the first two in the list are more important during term-time, while the remaining goals are more important during the vacations.)

3. Do you consider that resolving staff problems, requests for advice, requests for information, etc. is one of the goals of your subdivision? (Please tick one option)

(i) Yes

   (But as a System Manager, he would like to distance himself from it.)

4. If you answered “yes” to the previous question, could you indicate how important the goal is to your subdivision. (Please tick one option)

(iii) Highest importance

5. (a) What percentage of user problems, advice or information requests, etc. do you resolve on the spot while the user is with you? (Write down a percentage from 0% to 100%)

25%

(b) What percentage of user problems, advice or information requests, etc. do you pass on to another person or group? (Write down a percentage from 0% to 100%)

25%

(c) What percentage of user problems, advice or information requests, etc. do you add to your own list of problems to be investigated and resolved? (Write down a percentage from 0% to 100%)

320
50%
(of these, around 80% are resolved in 15 to 30 minutes, while the remainder take more than 30 minutes.)

6. I would like you now to consider how you manage those problems, etc. which you have added to your own list of problems to be investigated and resolved.

(a) Do you record the existence of the problem in some way e.g. writing down the details or do you rely on your memory? (Please tick one option)

(i) Record problem details (specify how)
Two methods are used:
(a) Problems are written down on post-it notes and attached to the USM’s workstation. Notes placed centrally on the screen have a higher priority than those placed on the side of the screen.
(b) Some problems are written down in a log book.

(b) If you have a list of long term problems to investigate and resolve, how do you decide in what order to tackle them? (E.g. do you assign a priority to them either explicitly or in your head?)

Each problem is assigned a priority based upon the number of people who are impacted by the problem: the higher the number of people impacted, the higher the priority assigned to the problem. The more that people complain, the more they get attention.

(c) On average, how long is your list of long-term problems?

12

(d) What happens to problems that do not seem possible to resolve?

Such problems do not tend to exist.

(e) Do you keep the user (problem owner) informed of the current status of a problem?

The user is informed when a problem is fixed (unless the user specified otherwise, or if further problems were encountered.)

(e) If “yes”, at what intervals?

(v) at significant events in problem resolution (if anything)

(f) Do you think there are any problems with the system that you use to manage your subdivision’s long term problems?
(a) The Unix SM is not certain that no problems are ever “lost”.
(b) The Unix SM feels the system is too reactive.
(c) The Unix SM feels the system is too sensitive to personal pressure.

7. In an ideal world, how would you like to improve your problem management system?

(a) Support for either an automatically-scheduled or self-scheduled stack of problems.
(b) Less involvement with users.
(c) Receive all problems via the helpdesk.
(d) Need to be able to cut off from users sometimes.

8. Is there anything else that you think I should know about problem management?

Problems recur periodically, e.g. yearly. It would be useful to have an archive of problems that was searchable with keywords. This could be used to refresh one’s memory of a problem and its solution, and to share knowledge with other staff, particularly new staff.

Questionnaire responses for the Small Systems Manager

1. High-level goals:

As the subdivision System Manager what do you consider are the main high-level goals of your domain of responsibility, the Helpdesk? For each goal can you supply the following information:

(a) A name and description for the goal.
(b) The name of the person or agency that assigned the goal to you, and the capacity in which they were acting. (This may include self-assigned goals)
(c) The kind of goal this is. (Increase, Maximize, Achieve, Maintain, Avoid, Cease from, Minimize, Decrease)
(d) The importance of this goal. Highest Importance, Important, Not Highest Importance

(a) Ensure usable laboratories.
   (a) Making sure that computer laboratories are usable by academic staff (highest priority) and students (next priority).
   (b) Job description
APPENDIX M. SUBSYSTEM OWNERS’ RESPONSES TO QUESTIONNAIRE

2. Stability of goals:

How frequently do your goals change (tick one of the following):

(b) Goals in my domain of responsibility are relatively stable.

3. Do you consider that resolving staff problems, requests for advice, requests for information, etc. is one of the goals of your subdivision? (Please tick one option)

(i) Yes

4. If you answered “yes” to the previous question, could you indicate how important the goal is to your subdivision. (Please tick one option)

(iii) Highest importance

5. (a) What percentage of user problems, advice or information requests, etc. do you resolve on the spot while the user is with you? (Write down a percentage from 0% to 100%)

75%

(b) What percentage of user problems, advice or information requests, etc. do you pass on to another person or group? (Write down a percentage from 0% to 100%)

0%

(c) What percentage of user problems, advice or information requests, etc. do you add to your own list of problems to be investigated and resolved? (Write down a percentage from 0% to 100%)

25%

(Often problems cannot be solved immediately because appropriate equipment is not available)

6. I would like you now to consider how you manage those problems, etc. which you have added to your own list of problems to be investigated and resolved.

(a) Do you record the existence of the problem in some way e.g. writing down the details or do you rely on your memory? (Please tick one option)

(ii) Rely on memory to retain problem details

(But, when busy, the SSM might make an index of problems from the problem list.)
APPENDIX M. SUBSYSTEM OWNERS’ RESPONSES TO QUESTIONNAIRE

(b) If you have a list of longterm problems to investigate and resolve, how do you decide in what order to tackle them? (E.g. do you assign a priority to them either explicitly or in your head?)

- the SSM tackles the most interesting (pleasurable) ones first.
- there are important problems (most tend to be equipment-related) that cannot be solved on the spot but are scheduled with a high priority; these are tackled whenever possible, e.g. when required equipment arrives.
- some problems are stored up for either the winter or summer vacations

(c) On average, how long is your list of long-term problems?

10

(d) What happens to problems that do not seem possible to resolve?

Such problems are placed on the “back-burner”. They may be restarted following opportunistic discovery, e.g. of a needed package on the net.

(e) Do you keep the user (problem owner) informed of the current status of a problem?

No

(e) If “yes”, at what intervals? Not applicable.

(f) Do you think there are any problems with the system that you use to manage your subdivision’s long term problems?

(a) Not keeping an explicit list of problems. This leads to:
   i. Forgetting problems.
   ii. Reduced ability to prioritise and organise problems.

(b) The SSM finds it hard to say “no” to users, i.e. to turn them away when they have problems.

7. In an ideal world, how would you like to improve your problem management system?

(a) The SSM would like to be e-mailed with problems or have them provided via a computer-based system. This is because he would like to control how he is made aware of problems. E.g. if problems arrive by e-mail he can examine them one at a time when he wants. This would
prevent the SSM having to deal with many users, all trying to report problems simultaneously.

(b) The SSM would like a computerised “personal assistant” to organise and prioritise the scheduling of tasks. But he would want to be able to override the order of scheduling e.g. to sometimes increase the priority of exciting tasks.

(c) Prioritise tasks.

(d) To have a means of notifying users when the SSM is available (or not) for receiving problems.

(e) Keep a permanent log of problems, for two reasons:
   • it would be pleasing to review what one had achieved
   • one could locate and reuse stored knowledge, e.g. telephone names, addresses, and numbers associated with solved recurring problems.
   • Reduce the number of problems to be dealt with in two main areas:
     – system failures might be sorted out faster if there were explicit procedures to be followed and whole backups kept
     – preventing students and staff from resetting machines

8. Is there anything else that you think I should know about problem management?

No data.

Questionnaire responses for the Oracle System Manager

The Oracle System Manager (OSM) shares her workload with a colleague. This is referred to in the data presented below. She refers in her responses to both problems and tasks. Tasks are what have to be done in order to tackle problems.

1. High-level goals:

As the subdivision System Manager what do you consider are the main high-level goals of your domain of responsibility, the Helpdesk? For each goal can you supply the following information:

(a) A name and description for the goal.

(b) The name of the person or agency that assigned the goal to you, and the capacity in which they were acting. (This may include self-assigned goals)

(c) The kind of goal this is. (Increase, Maximize, Achieve, Maintain, Avoid, Cease from, Minimize, Decrease)

(d) The importance of this goal. Highest Importance, Important, Not Highest Importance
APPENDIX M. SUBSYSTEM OWNERS’ RESPONSES TO QUESTIONNAIRE

(a) Introduce NT to faculty.
   (a) Introduce NT to faculty.
   (b) Assigned by colleague
   (c) Achieve
   (d) Highest-Importance
(b) Keep Oracle up to date
   (a) Keep Oracle up to date with new releases
   (b) Self-assigned
   (c) Maintain
   (d) Important
(c) Install NT
   (a) To install NT on machines used by the Administration team
   (b) Self-assigned
   (c) Achieve
   (d) Not Highest Importance

2. Stability of goals:

   How frequently do your goals change (tick one of the following):

   (b) Goals in my domain of responsibility are relatively stable.

3. Do you consider that resolving staff problems, requests for advice, requests for information, etc. is one of the goals of your subdivision? (Please tick one option)

   (i) Yes

4. If you answered “yes” to the previous question, could you indicate how important the goal is to your subdivision. (Please tick one option)

   (iii) Highest importance

5. (a) What percentage of user problems, advice or information requests, etc. do you resolve on the spot while the user is with you? (Write down a percentage from 0% to 100%)

   85%

   (b) What percentage of user problems, advice or information requests, etc. do you pass on to another person or group? (Write down a percentage from 0% to 100%)

   1%

   (c) What percentage of user problems, advice or information requests, etc. do you add to your own list of problems to be investigated and resolved? (Write down a percentage from 0% to 100%)

   14%
6. I would like you now to consider how you manage those problems, etc. which you have added to your own list of problems to be investigated and resolved.

(a) Do you record the existence of the problem in some way e.g. writing down the details or do you rely on your memory? (Please tick one option)

(i) Record problem details (specify how)

Three methods are used:

(a) For NT-related problems: the OSM uses a spreadsheet with two main sections; one section records user support related problems, the other records development related problems. The following categories of information are recorded for each problem:
- Problem name
- Problem description
- Machine name
- Machine owner name
- Problem priority
- Problem finished flag
- Comments

(b) For Unix-related problems: the OSM maintains problems in a “to do” list (one sentence per problem) stored as an Emacs file in the order in which they arrive.

(c) For PC-related problems: the OSM maintains a log in which she records notes on the problem. She adds new information to the log as and when it is acquired.

This log is also used to record details of phone calls related to NT, Unix, or PC problems.

(b) If you have a list of longterm problems to investigate and resolve, how do you decide in what order to tackle them? (E.g. do you assign a priority to them either explicitly or in your head?)

The following criteria are used:

(a) tackle first those problems that affect the most people;
(b) tackle first those problems deemed most urgent by the reporting users (after adjusting the user’s evaluation, if necessary);
(c) tackle first irritating problems;
(d) tackle high status users’ problems first (but not always);
(e) tackle first short tasks (perhaps in parallel with longer tasks);
(f) Tackle first those problems that have remained unsolved the longest.

(c) On average, how long is your list of long-term problems?

20

(d) What happens to problems that do not seem possible to resolve?

(a) Important problems: get in touch with the associated proprietor’s relevant support department.
(b) Minor problems: These go to the “bottom of the heap”, i.e. they are assigned the lowest priority. This often occurs after some sort of fix or workaround has been installed. Opportunistic discovery of relevant, new information may trigger a review of such a problem.

(e) Do you keep the user (problem owner) informed of the current status of a problem?

Yes

(e) If “yes”, at what intervals?

(v) at significant events in problem resolution

(f) Do you think there are any problems with the system that you use to manage your subdivision’s long term problems?

(a) The three different problem management systems (NT, Unix, PC) that are used are not integrated.
(b) One cannot “see” what other people working on related tasks are doing, e.g. whether a problem has been received (and possible reported) by another Manager, since there is not a global COTS system.
(c) There are problems specific to each of the three systems:
   i. NT problem management system (spreadsheet):
      A. The spreadsheet is slow to load up.
      B. It is often time-consuming to access the right place to input a new problem (sometimes six or seven pages need to be paged through).
   ii. Unix problem management system (Emacs “to do list” file):
      A. This system does not currently store enough information.
      B. This system does not encourage the storing of all required information, i.e. it does not actively prompt for information (process guidance)
APPENDIX M. SUBSYSTEM OWNERS’ RESPONSES TO QUESTIONNAIRE

or passively prompt for information (information category).

iii. PC problem management system (logbook):
   A. Pages are easily lost.
   B. Pages are unformatted.

7. In an ideal world, how would you like to improve your problem management system?

   (a) There should be a global system.
   (b) My three systems should be integrated.
   (c) It should be easy to access information, carry out searches, and perform ad hoc querying.
   (d) There should be an automated reminder of flagged tasks.
   (e) It should be possible for me to schedule and reschedule tasks.
   (f) I would prefer to receive problems via the phone or by e-mail rather than face-to-face as it is easier to schedule those received in the former manner. (However, face-to-face communication would still be needed.)
   (g) I would like more academic staff to channel their problems through the Helpdesk rather than through the System Managers.

8. Is there anything else that you think I should know about problem management?

   • 75% of the job is taken up with ongoing resourcing and maintenance of NT systems.
   • Problems need to be separated into user problems and development problems.
   • It is difficult to force oneself to do some development tasks even though they are the same order of priority, because they take longer.

Questionnaire responses for the PC System Manager

1. High-level goals:

   As the subdivision System Manager what do you consider are the main high-level goals of your domain of responsibility, the Helpdesk? For each goal can you supply the following information:

   (a) A name and description for the goal.
(b) The name of the person or agency that assigned the goal to you, and the capacity in which they were acting. (This may include self-assigned goals)

(c) The kind of goal this is. (Increase, Maximize, Achieve, Maintain, Avoid, Cease from, Minimize, Decrease)

(d) The importance of this goal. Highest Importance, Important, Not Highest Importance

(a) Provide a stable, reliable PC network
   (a) Provide a stable, reliable PC network
   (b) Job description
   (c) No data
   (d) Highest-Importance

(b) Provide support for PC applications
   (a) Provide support for PC applications
   (b) Job description
   (c) No data
   (d) Important

(c) Provide PC-related support for staff and students
   (a) Provide PC-related support for staff and students
   (b) Job description
   (c) No data
   (d) Important/Highest Importance

(d) Balancing pressures for access to PC facilities against maintaining a reliable infrastructure.
   (a) Balancing pressures for access to PC facilities against maintaining a reliable infrastructure
   (b) Job description
   (c) No data
   (d) Important/Highest Importance

(e) Providing a system compatible with the demands placed upon it (i.e. one that uses established rather than state of the art technology).
   (a) Providing a system compatible with the demands placed upon it (i.e. one that uses established rather than state of the art technology)
   (b) Job description
   (c) No data
   (d) Important

(f) Looking at long-term picture (one year ahead) and acting accordingly
   (a) Looking at long-term picture (one year ahead) and acting accordingly
   (b) Job description
(c) No data
(d) Important

2. Stability of goals:

How frequently do your goals change (tick one of the following):

(b) Goals in my domain of responsibility are relatively stable.

3. Do you consider that resolving staff problems, requests for advice, requests for information, etc. is one of the goals of your subdivision? (Please tick one option)

(i) Yes

4. If you answered “yes” to the previous question, could you indicate how important the goal is to your subdivision. (Please tick one option)

(iii) Highest importance

5. (a) What percentage of user problems, advice or information requests, etc. do you resolve on the spot while the user is with you? (Write down a percentage from 0% to 100%)

90%

(b) What percentage of user problems, advice or information requests, etc. do you pass on to another person or group? (Write down a percentage from 0% to 100%)

1%

(c) What percentage of user problems, advice or information requests, etc. do you add to your own list of problems to be investigated and resolved? (Write down a percentage from 0% to 100%)

9%

6. I would like you now to consider how you manage those problems, etc. which you have added to your own list of problems to be investigated and resolved.

(a) Do you record the existence of the problem in some way e.g. writing down the details or do you rely on your memory? (Please tick one option)

(i) Record problem details (specify how)

Three methods are used:

(a) An offline flippad is used for recording problem details in an unformatted way. The problem details are crossed out when the job is completed (i.e. when the problem is resolved).
(b) An online jobs database is used for longer-term problems. This comprises a high-level index of jobs where each entry is a short meaningful phrase. The index points to an area where free-form problem information is recorded. The information might include a problem description, progress on the task (dated), details of related problems, and status (“A” means active, “B” means ongoing, and “C” means complete).  

(c) A system for tracing problem files using a keyword search.  

This log is also used to record details of phone calls related to NT, Unix, or PC problems.

(b) If you have a list of long-term problems to investigate and resolve, how do you decide in what order to tackle them? (E.g. do you assign a priority to them either explicitly or in your head?)

The following criteria are used:

(a) tackle first those problems where something has stopped working;

(b) tackle first those problems that affect the most people;

(c) tackle first those problems where there is a known “quick fix”—so if there is a known “quick fix” the PCSM would tend to apply it immediately.

(c) On average, how long is your list of long-term problems?

10—20 of which 5—6 might be critical.

(d) What happens to problems that do not seem possible to resolve?

(a) Some eventually disappear;

(b) Some solve themselves, e.g. the problem owner leaves;

(c) Some remain on the “back burner”.

(e) Do you keep the user (problem owner) informed of the current status of a problem?

Depends on the scale of the problem. Yes, if network problem.

(e) If “yes”, at what intervals?

(v) at significant events in problem resolution

(f) Do you think there are any problems with the system that you use to manage your subdivision’s long-term problems?
(a) Prioritisation of problems is ad hoc: one can never “know” if the right decision was made;
(b) problems “drop off” the list of problems;
(c) no problem-related statistics are kept;
(d) in particular problem resolution times are not recorded;
(e) it is easy to forget or overlook the recording of important information.

7. In an ideal world, how would you like to improve your problem management system?

(a) There should be more information available “on tap”. For example netwire magazine FAQ should be available online or in a hardcopy form;
(b) More time should be made available for strategic work by drawing firmer boundaries around office responsibilities. For example, more user support should be given to junior team members.
(c) There could be a centralised system with problems held in tables on Unix. This data could be accessed by a “reader” on a PC that looks at data but does not copy it across (like a telnet connection). Access to such a centralised system should be possible within 10 seconds.

8. Is there anything else that you think I should know about problem management?

No data.

Questionnaire responses for the Unix Computer Officer

1. High-level goals:

As the subdivision System Manager what do you consider are the main high-level goals of your domain of responsibility, the Helpdesk? For each goal can you supply the following information:

(a) A name and description for the goal.
(b) The name of the person or agency that assigned the goal to you, and the capacity in which they were acting. (This may include self-assigned goals)
(c) The kind of goal this is. (Increase, Maximize, Achieve, Maintain, Avoid, Cease from, Minimize, Decrease)
(d) The importance of this goal. Highest Importance, Important, Not Highest Importance
APPENDIX M. SUBSYSTEM OWNERS’ RESPONSES TO
QUESTIONNAIRE

(a) Provide Unix education
   (a) Educate users to use Unix systems efficiently
   (b) Self-generated
   (c) maximise
   (d) Highest-Importance

(b) To provide a good, working Unix environment
   (a) To provide a good, working Unix environment
   (b) Self-generated
   (c) Maintain and maximise
   (d) Important

(c) Learn as much as possible about programming languages
   (a) Learn as much as possible about programming languages
   (b) Self-generated
   (c) Maximise
   (d) Important

(d) Promote cooperative working within the faculty
   (a) Promote Computer Officers and academics and users and management working together. infrastructure
   (b) Self-generated
   (c) Maximise
   (d) Not Highest Importance (unrealistic)

2. Stability of goals:

   How frequently do your goals change (tick one of the following):

   (b) Goals in my domain of responsibility are relatively stable.

3. Do you consider that resolving staff problems, requests for advice, requests for information, etc. is one of the goals of your subdivision? (Please tick one option)

   (i) Yes

4. If you answered “yes” to the previous question, could you indicate how important the goal is to your subdivision. (Please tick one option)

   (i) Not very important

5. (a) What percentage of user problems, advice or information requests, etc. do you resolve on the spot while the user is with you? (Write down a percentage from 0% to 100%)

   80%
APPENDIX M. SUBSYSTEM OWNERS’ RESPONSES TO QUESTIONNAIRE

(b) What percentage of user problems, advice or information requests, etc. do you pass on to another person or group? (Write down a percentage from 0% to 100%)

10%

It depends upon who is staffing the helpdesk and how well they know what are appropriate problems for the Unix CO.

(c) What percentage of user problems, advice or information requests, etc. do you add to your own list of problems to be investigated and resolved? (Write down a percentage from 0% to 100%)

10%

6. I would like you now to consider how you manage those problems, etc. which you have added to your own list of problems to be investigated and resolved.

(a) Do you record the existence of the problem in some way e.g. writing down the details or do you rely on your memory? (Please tick one option)

(i) Record problem details (specify how)

Four methods are used:

(a) E-mail: the Unix CO’s mailbox acts as a database of problems. These can include e-mails of problems sent by the Unix CO to herself.

(b) Helpdesk sheets: the unix CO maintains a pile of helpdesk sheets, processes them, and returns each to the Helpdesk Manager when the associated problem has been resolved.

(c) Meeting minutes: tasks that are assigned in meetings (as minutes) can act as a list of problems to be resolved.

(d) To do list.

(b) If you have a list of longterm problems to investigate and resolve, how do you decide in what order to tackle them? (E.g. do you assign a priority to them either explicitly or in your head?)

The following criteria are used:

(a) tackle first those problems that affect the most people;
(b) tackle first those problems owned by part-timers;
(c) tackle first those problems reported face-to-face.

(c) On average, how long is your list of long-term problems?

No data.
APPENDIX M. SUBSYSTEM OWNERS’ RESPONSES TO QUESTIONNAIRE

(d) What happens to problems that do not seem possible to resolve?

(a) They tend to fade away because they are placed on the “back burner”.
(b) Some are taken to other Computer Officers.
(e) Do you keep the user (problem owner) informed of the current status of a problem?

It depends.
(e) If “yes”, at what intervals?
(v) at significant events in problem resolution
(f) Do you think there are any problems with the system that you use to manage your subdivision’s long term problems?

(a) It is too ad hoc;
(b) It is interrupt driven;
(c) Some long-term tasks are not started.

7. In an ideal world, how would you like to improve your problem management system?

(a) Reduce number of problems (to zero);
(b) Only work on long-term tasks;
(c) Work without interrupts;
(d) Receive tasks in a controlled way, e.g. at certain times of the day and only by e-mail
(e) Be able to put related tasks together;
(f) Have scheduled times for interrupt and interrupt-free modes of working.

(g)

8. Is there anything else that you think I should know about problem management?

• Documentation is poor;
• Students lack basic computing knowledge.

Questionnaire responses for the PC Computer Officer

1. High-level goals:

As the subdivision System Manager what do you consider are the main high-level goals of your domain of responsibility, the Helpdesk? For each goal can you supply the following information:
APPENDIX M. SUBSYSTEM OWNERS’ RESPONSES TO QUESTIONNAIRE

(a) A name and description for the goal.
(b) The name of the person or agency that assigned the goal to you, and the capacity in which they were acting. (This may include self-assigned goals)
(c) The kind of goal this is. (Increase, Maximize, Achieve, Maintain, Avoid, Cease from, Minimize, Decrease)
(d) The importance of this goal. Highest Importance, Important, Not Highest Importance

(a) Ensure all laboratory PCs are “up and running”.
   (a) Ensure all laboratory PCs are “up and running”.
   (b) PC System Manager
   (c) Maintain
   (d) Highest-Importance

(b) Ensure all PC software is available on laboratory PCs
   (a) Ensure all PC software is available on laboratory PCs
   (b) PC System Manager
   (c) Achieve
   (d) Highest-Importance

(c) Ensure that all academic PCs are “up and running” and that all the appropriate PC software is available
   (a) Ensure that all academic PCs are “up and running” and that all the appropriate PC software is available
   (b) PC System Manager
   (c) Maintain
   (d) Highest Importance

(d) Promote cooperative working within the faculty
   (a) Investigate new PC hardware and software technologies with a view to bringing them in-house
   (b) Job description
   (c) Maintain
   (d) Important

(e) Act as a backup for the PC System Manager when appropriate.
   (a) Act as a backup for the PC System Manager when appropriate.
   (b) PC System Manager
   (c) Maintain
   (d) Not Highest Importance

2. Stability of goals:

   How frequently do your goals change (tick one of the following):

   (b) Goals in my domain of responsibility are relatively stable.
3. Do you consider that resolving staff problems, requests for advice, requests for information, etc. is one of the goals of your subdivision? (Please tick one option)

   (i) Yes

4. If you answered “yes” to the previous question, could you indicate how important the goal is to your subdivision. (Please tick one option)

   (iii) Highest importance

5. (a) What percentage of user problems, advice or information requests, etc. do you resolve on the spot while the user is with you? (Write down a percentage from 0% to 100%)

   90%

   (b) What percentage of user problems, advice or information requests, etc. do you pass on to another person or group? (Write down a percentage from 0% to 100%)

   1%

   It depends upon who is staffing the helpdesk and how well they know what are appropriate problems for the Unix CO.

   (c) What percentage of user problems, advice or information requests, etc. do you add to your own list of problems to be investigated and resolved? (Write down a percentage from 0% to 100%)

   9%

6. I would like you now to consider how you manage those problems, etc. which you have added to your own list of problems to be investigated and resolved.

   (a) Do you record the existence of the problem in some way e.g. writing down the details or do you rely on your memory? (Please tick one option)

   (i) Record problem details (specify how)

       Four methods are used:

       (a) Remembers problems which can be fixed within a short time;

       (b) Records details on unformatted note-pad, e.g. for systems development;

       (c) Asks staff to e-mail details of problems;

       (d) Shares the Oracle Support Managers method of recording NT problems.

   (b) If you have a list of longterm problems to investigate and resolve, how do you decide in what order to tackle them? (E.g. do you
assign a priority to them either explicitly or in your head?)

The following criteria are used:
(a) Tackle first those problems having the most severe effect on the reporting individual.
(c) On average, how long is your list of long-term problems?

No data.

(d) What happens to problems that do not seem possible to resolve?

No data.

(e) Do you keep the user (problem owner) informed of the current status of a problem?

Yes.
(e) If “yes”, at what intervals?
(v) at significant events in problem resolution

(f) Do you think there are any problems with the system that you use to manage your subdivision’s long term problems?

(a) Tracking where a problem has got to, particularly if it is necessary to go outside of the faculty for help;
(b) Tracking problems when they become the responsibility of more than one COTS team;
(c) It is not possible to produce precise statistics on the breakdown of work completed. For example it is not possible to find out exactly how much time is taken up by non-CSM students.

7. In an ideal world, how would you like to improve your problem management system?

(a) There could be a centralised problem management system based on the web - it would need to be fast and easy to use;
(b) students should log faults and problems to a central database and be e-mailed with the results of queries;
(c) the centralised system should be able to produce statistics on user-problem solving;
(d) users should be able to book times to meet Computer Officers;
(e) students and staff should be made aware of what COs can and cannot do within different timescales.

8. Is there anything else that you think I should know about problem management?
• Currently the kinds of problem reported is not monitored.
• Currently the usage of different packages is not monitored.
Appendix N

Observing the COTS user-problem management system
Introduction

In October 1997, I spent six hours closely observing and recording the activities of the three COTS groups that constitute the problem management system: the Helpdesk, User Support and System Support (two hours per group). In addition, I asked these staff questions about their work. This observational and questioning activity was an attempt by me to incorporate the essence of the Contextual Design [HB96] method into the approach being synthesised. It was hoped that analysis of both these observations and the answers to the questions would lead to insights into the kind of CBS (and thus to its requirements) that would and would not support these staff in their work.

The results from this work is detailed in the following three sections. Each section is divided into four parts. First, the context of the work is described. Second, the detailed observations of the work over time are presented. Third, an analysis of these observations is made. And finally, the implications for possible supporting CBSs (and thus for their requirements) are brought out.

Observing the COTS user-problem management system

Helpdesk

Context

On Tuesday 21st of October 1997, I spent two hours, from 1000h to 1200h, observing the activities that Helpdesk personnel performed. Personnel during that period included Liz Davies, the Computer Officer in charge of the Helpdesk, Karen, a Help Assistant, who worked from 1000h until 1130h, and Maria, also a Help Assistant, who worked from 1130h until 1200h.

Throughout most of this period, Liz tended to be at the back of the Helpdesk area, working in her open-plan office at management tasks: ordering consumables - books, discs, handouts, etc.; entering new books into the library system, and so on. From time to time, she came to the Helpdesk counter to deal directly with a user’s query. She indicated to me that it was her policy to focus on management tasks, but to help the Help Assistant when they were under pressure, i.e. when there were queues of users.

From time to time during my observation, I asked Liz Davies, Karen and Maria, for their view on various aspects of a computer-based problem management system. These events are included below using italics.

Observations
Karen and Liz are preparing to open the Helpdesk: Liz is checking the money in the till. Jin Sa (JS) (staff) asks LD who is in charge of Unix mail and who is in charge of PC mail. LD sends JS to see Philip Naylor (PN) (Acting Principal Computer Officer).

Five student (S1 - S5) ask Karen (K) about the availability of a WordPerfect Office Mail document. K hands out one copy of the document to each student.

S6 tells K he would like to see Steve Allman (SA) (Networks Computer officer). SA is located in an office close by but out of sight. K tells him that SA is on the 'phone. K advises S6 to come back later. S6 says that he will come back later.

K checks Help Assistants’ availability on an online spreadsheet.

LD retires to her office to work on management tasks.

LD takes money to Clare Ebdon (CE) (Faculty Finance Administrator). CE works one floor down at the other end of the building. K checks that printers have paper in them.

Printer prints. S7 tells K he would like to see Steve Allman (SA). K checks SA’s availability. K tells S7 that SA is on the ‘phone. K offers to take a message from S7 for SA. S7 says he is looking for material ordered for a 3rd year project. S7 says that he will come back later.

S8 asks K for 2 printouts. S8 says that he selected the colour option so he expects coloured listings K bring 2 black and white printouts. S8 asks how he can use a colour printer for a Word document. K says she does not know. K advises the student to come back later when LD is available because LD will be able to advise him.

S9 asks K how to print postscript files from home. K say that she does not know. Staff member asks if K has any printouts for him. K says no. Staff member says he printed them 20 minutes previously. Staff member goes away. K checks "Help" leaflets and restocks some of them. Printer prints.

K checks whether SA is still on the ‘phone. (SA is still on the ‘phone). K makes us tea.

K checks whether SA is still on the ‘phone. K checks Help Assistants’ availability on an online spreadsheet.

S10 asks if K has driver disc for parallel port CD rom drive. K says no. K advises S10 to try Rob Williams (staff) or Craig Duffy (staff). LD returns. K checks whether SA is still on the ‘phone. K clears down the printers. Stephen Mullen (SM) (Assistant Computer Officer, User Support(Unix)) asks K about a listing just printed.
S11 collects and pays K 10p for laser printout. K and LD discuss colour printing. K and LD agree a price of 15p per page. K asks LD how can students do colour printing. LD tells K that it must be done from a machine in 3P12 (i.e. in the room where the Helpdesk area is located). Staff Research Assistant (SRA) asks K how can she find a CSM student’s email address to send mail to them using Elm. K suggests SRA use finger. K shows SRA Elm Helpsheet. (But it doesn’t help.) K sends SRA to User Support. (SM helps SRA to find required email address.)

S12 tells K about his need to obtain from SA a CD disc for installing Linux. It should be in the CD player but is not there. K goes to see SA. S12 waits. K tells S12 that SA is searching for it.

S13 pays for and collects printout. S14 pays for and collects printout. S15 asks K for “hole-puncher”. K gives him the “hole-puncher”. S16 asks K whether he can printout a portion of the “home page”. K says she does not know. K tells him that there is no-one available at the moment who can help him. K tells him that Julia Dawson (JD) would normally be able to help him. K checks whether SA has returned.

K gives SM some printouts. I ask K how feasible it would be for her to type in details of problems if a computer-based problem management system were used on the Helpdesk. K says that she would find typing in a lot of details to be too time-consuming. And that this would be particularly true when she was busy, for example during the lunch hour. Bob Lang (staff) inquires whether the Helpdesk support a lost-property service as he has lost a textbook on Java. Margaret Clements (staff) asks LD about the possibility of managing material for the Mathematics Learning Centre. K asks SA about the CD Rom problem. S17 asks K for a pen. K gives S17 a pen. K says, after I suggest it to her, that she wouldn’t mind pressing one or two keys for each interaction. She indicates that there might be a problem at shift change-over. (Is this because the Help Assistants cannot wait around after a change-over because e.g. they must go to lectures?) S18 asks K how to contact Glyn Watkins (staff) K tells S18 to inquire again in 2P50, the Faculty Office where they keep such information.

S19 scans helpsheets and takes two. S19 asks K whether the network printer is ok. K checks this with User Support. K tells S19 that it’s ok.

A staff member asks K whether PN (System Support) is in. K checks PN’s office. K tells staff member that PN is in and shows her into his office. (PN later transfers her to User Support.) SM (User Support) collects printouts form K. Staff member goes directly to Systems Support to see SA. S20 goes directly to User Support.
1105  S21 asks to collect printout. K asks S21 his printer is set up for letter format or for A4. (S21’s printouts have one word per page) S22 arrives. (No Helpdesk personnel free) S22 waits. S22 leaves. K tells S21 to set his printer up for letter format not for A4. S22 asks K to place in a box a disc for Chris Reynold’s project. K complies.

1115  S22 asks K how he can format a hard-drive disc. K says she does not know. S22 asks her to ask someone. K asks LD. LD tells K. K tells S22 how to do it: “put boot disc in...etc.” S22 pays for printout. (Now correctly formatted.)

1117  S22 goes directly to System Support. LD answers ‘phone. (She is asked to supply the email address of a member of staff.) LD supplies the requested email address. S23 asks if BURK’s disc is in. LD says no and that she will put up notices when it is in. S23 pays for and collects printout. Frank Maddix (staff) goes directly to User Support. K indicates, after I suggest it to her, that typing in a user’s system id, while using a computer-based problem management system might be feasible. The id could be used to obtain more complete user details.

1125  S24 asks for a BASA project. K helps S24 to look for the project in the project folder. K asks LD about BASA projects. LD tells K where more BASA projects are stored.

1130  K ends her shift. Maria (M) (Help Assistant) starts her shift. S24 takes BASA project and signs for it with LD. S25 requests Unix and PC passwords. M asks LD about the availability of PC passwords. M takes S25 to User Support. S26 asks LD whether Swipcsard photographs are available. LD tells him that Swipcsard photo is not the same as CSM photo. LD advises him to email JD and ask for CSM photo. LD says that a dedicated machine is necessary to support the Helpdesk library system. She asks how a computer-based problem management system would benefit the Helpdesk in particular. I suggest the need to prototype any putative CBPMS.

1145  M answers a call to LD from a rep. concerning consumables.

1150  S25 tell M about a problem starting Windows95. M says she had same problem and that he needs to install Windows 3.1 first. S25 asks M whether he should format the hard-drive. M takes this inquiry to User Support. M invites student through to User Support (To talk to Alan Price.)

1152  M clears down the printers. M also suggests that to type in a handful of key strokes would be feasible. M suggests that Helpdesk personnel could make notes by hand during/after a transaction. These could be entered into the system at a later (quieter) time. S26 asks which way up do slides go in the photocopier. M asks LD. LD cannot remember. LD suggests the student tries various ways. S27 asks to see AP.

1158  S27 goes in to see AP. LD adds price of transparency to yellow folder. Member of staff collects printout.

1200  M ends her shift.
Table N.1: Summary of problems encountered in the observation period

<table>
<thead>
<tr>
<th>Information type</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problems</td>
<td>30</td>
</tr>
<tr>
<td>Problems resolved by Helpdesk team</td>
<td>21</td>
</tr>
<tr>
<td>Problems resolved by Helpdesk team and System Support</td>
<td>3</td>
</tr>
<tr>
<td>Problems passed to User Support</td>
<td>4</td>
</tr>
<tr>
<td>Problems unresolved</td>
<td>2</td>
</tr>
</tbody>
</table>

Analysis

Table N.1 shows how requests for services were dealt with during the two-hour observation period. The services included solving technical problems, supplying information, supplying printouts, etc.

The table shows that two thirds of service requests were resolved by Helpdesk staff. It also shows that almost one quarter of requests were passed to colleagues in User Support and Systems Support.

I was interested to see whether it would be worthwhile for a record of all service requests to be recorded by the Helpdesk team in any Computer-Based Problem Management System. However, it seems clear that this would not be worthwhile functionality, since the majority of service requests are resolved quickly on-the-spot. (It may be the case that User Support and Systems Support should record details of service requests that are passed to them by the Helpdesk team. By that stage, such service requests are likely to be mainly technical problems experienced by users (staff and students)).

Besides the fact that it is probably unnecessary for the Helpdesk team to record details of all service requests, various members of this team indicated that it would be difficult in any case to make such a record. For example, K says that she would find typing in lots of details too time consuming.

Although it may not be necessary to record details of all service requests, it may be useful (e.g. for management purposes) to have a record of the number of service requests of each type that occur in each period. This could be achieved, for each request, by clicking one key stroke. K says, after I suggest it to her, that she wouldn’t mind pressing one or two key strikes for each interaction with a user.

In addition to the observations recorded in the table above, I also noted that there were four instances of “illegal” user behaviour. On these occasions, either a member of staff or a student went directly to User Support or to Systems Support without first contacting a member of the Helpdesk team. However, it does not seem necessary to tackle this problem via a CBPMS; a simple barrier system operated by the Helpdesk team that controls access to User Support and Systems Support should suffice.

I also noted two other Helpdesk problems. First, it is difficult to understand what some problem owners are saying. This might be because their speech is characterised by computing jargon, because they are foreign and do not speak English well, because their accents obscure what they are say-
ing, because they speak very quickly, because they mumble, or for various combinations of these reasons.

Second, some problem owners adopt an arrogant manner towards the Helpdesk personnel.

Both these features make it more difficult than it would otherwise be for Helpdesk staff to assist problem owners. But it is difficult to see how a CBPMS could successfully resolve or attenuate these.

**CBPMS requirements**

The following requirements or non-requirements for a CBPMS resulted from the observation/questioning activity.

1. The CBPMA should not force the Helpdesk team to enter problem descriptions.

2. The CBPMS should provide support for allowing a member of the Helpdesk team (or similar) to record the type of each service request quickly and easily.

**User Support**

**Context**

On Tuesday the 21st of October 1997, I spent two hours, from 1400h to 1600h, observing the activity of Stephen Mullen, a member of the user Support Group. Stephen shared an office with three other members of the group. The office was located a few yards from the Helpdesk area.

**Observation**
APPENDIX N. OBSERVING THE COTS USER-PROBLEM MANAGEMENT SYSTEM

1400 Stephen Mullen (SM) is helping David Forsyth (staff), who has a password problem. Liz Davis has a query for SM. SM is currently working on a Postscript Printer problem. The printer has been "down" for twenty minutes. Alan Price (AP), a User Support colleague, asks SM if he looks after IP addresses. SM checks the printer.

1405 DF still cannot login on a Unix machine in room 2P10. SM invites DF to type in his password. SM gives DF a new password and a new user name.

1410 DF still cannot login. SM goes to lab. 3P10 and cannot login. SM returns to the User Support office, where it's working ok. SM suggests using 3P27 to DF. SM tries 3P9: it worked.

1415 Maria from the Helpdesk brings in a student with a quota problem. SM tries various things on a Unix machine. SM looks for large files to delete.

1420 SM is still working on the quota problem.

1425 SM answers the phone and looks for Sam. SM takes message. SM is still working on the quota problem.

1430 SM is still working on the quota problem.

1434 SM answers the phone and transfers the call to AP.

1435 SM solves the quota problem: a cache file overflow occurred through the use of Netscape.

1440 SM selects from e-mail list (of problems) the most urgent background problem to work on next: in this case it is restoring someone's e-mail files.

1445 SM goes to restore room in 2p46

1450 SM suspects that the wrong tape is loaded.

1452 Alwyn Barry interrupts SM with a problem; he needs to restore a student's directory. AB leaves details with SM. AB indicates that it is not an urgent problem. It can be resolved "whenever". SM puts down on his desk the piece of paper containing the details of this problem.

1454 SM returns to the original (restore) problem; it is difficult to resolve because the backup tapes are not in an optimal order.

1500 SM still working on the (restore) problem.

1504 SM seeks advice from Phil Naylor a colleague in Systems Support.

1506 PN confirms that the right machine is being used. SM thins that he is still not using the right tape.

1511 SM tries the tape in the machine downstairs in 2D46 - it is the last chance to find the right tape.
1517   SM still working on the restore problem.
1521   SM answers a call from a maintenance company.
1524   Karen (from the Helpdesk) brings in a student with a query, but SM is still on the 'phone. Karen asks the caller to return. SM checks the make and model of a machine for the caller.
1527   'Phone call ends
1527   Error message: cannot read tape
1528   Julia Dawson (JD) brings in a user-problem with photographs. SM says: "leave it with me". SM puts this problem to one side; he tells me that he will do it next because it won't take long.
1532   HK asks SM for PN's extension. HK asks SM to read a word.
1534   SM returns to the "restore" problem. SM goes to PN, in Systems Support, for advice, but PN is not in.
1536   Student cannot enter his password to unfreeze the screen on a Resolv machine. Student and SM confer with AP on this problem. AP solves this problem.
1538   SM tackles JD's photograph problem, but finds that he needs a book that is at home, so he suspends this problem.
1541   Back to the "restore" problem. SM e-mails the problem owner that this restore cannot be performed. The restore problem is thus terminated.
1544   SM selects the disc space problem from his e-mail list of problems to work on. (There are only two or three other problems on this list; one will be time-consuming so SM will leave it 'til tomorrow; the others are not very urgent.)
1550   SM solves the disc space problem. SM writes to this problem's owner.
1552   SM tackles the next problem and solves it immediately. SM writes to this problem's owner.
1555   Student requests user id and password. SM attends to this request straight away. SM gives the student printout details. This problem is resolved and terminated.
Analysis

During the two-hour period of observation, SM tackled problems from two sources: his long-term problem list and problem interrupts. When he was able to, SM selected a problem to work on from his long-term problem list, which he maintained as a list of e-mails (one e-mail per problem, generally). Each time that he was able to, he would select the next long-term problem to work on, in a dynamic way according to two main criteria: urgent problems were tackled first, but a problem was only chosen if SM deemed that there was a sufficiently large block of time available to him to resolve it in.

SM was frequently interrupted in his work. There were four main types of interrupt: Helpdesk personnel bringing students and their problems; User Support colleagues bringing problems; “illegal” direct approaches by students or staff; and external sources, e.g. a Maintenance Company. SM used the following strategy for dealing with interrupts. He would first ascertain the nature of the problem. If he judged that he could solve it quickly, then he would tackle it straight away; otherwise he would write the details of the problem on a scrap of paper and tackle it later. Effectively these latter problems were being added to his long-term problem list. I noted that scribbling problem descriptions on scraps of paper was perhaps not the best way of recording them: a scrap of paper might easily be mislaid, for example.

The two tables (Table N.2 and Table N.3) below summarise the key features of the two-hour period of observation. They summarise the problems brought to SM, and show the outcome for each problem. It is worth noting that SM tackled ten problems in the two hours. This compares with the thirty tackled by the Helpdesk over a similar period. Since the Helpdesk filter out many of the easier-to-solve problems, one would expect that SM would tackle fewer problems, but that they would be harder problems. Seven of the ten problems tackled by SM were interrupt problems, while three were from his long-term problem list. SM resolved seven of the ten problems brought to him; two were added to his long-term problem list; and one student was told to return again later.

CBPMS requirements

SM did not think that a Computer-Based Problem Management System would be useful to him. SM prefers paper-based descriptions of problems. He doubted whether problem information could be collected electronically.

However, as the observer, I felt that it would be useful for SM to have a way of recording problem details using a CBPMS. Problem descriptions would be much less likely to be lost if recorded in a CBPMS than if written on scraps of paper. I noted that his long-term problem list (e-mails) was already electronically-based. I inferred from this the following requirements for a CBPMS:

1. The CBPMS should provide support for displaying a list of problems. Each problem should be associated with a priority.

2. The CBPMS should provide support for quickly and easily entering the description of a problem.
Table N.2: Source of problems encountered in the observation period and their resolution

<table>
<thead>
<tr>
<th>Problem identifier</th>
<th>From problem list</th>
<th>Staff interrupts</th>
<th>Student interrupts</th>
<th>Added to problem list</th>
<th>Resolved</th>
<th>Told to return</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>quota</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>file</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>disc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>query</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>photo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>screen</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>disc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>next</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>user id</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>3</strong></td>
<td><strong>3</strong></td>
<td><strong>4</strong></td>
<td><strong>2</strong></td>
<td><strong>7</strong></td>
<td><strong>1</strong></td>
</tr>
</tbody>
</table>

Table N.3: Summary of problems encountered in the observation period

<table>
<thead>
<tr>
<th>Information type</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problems</td>
<td>10</td>
</tr>
<tr>
<td>Problems selected from long-term problem list</td>
<td>3</td>
</tr>
<tr>
<td>Problems resolved by User Support</td>
<td>7</td>
</tr>
<tr>
<td>Problems passed to Systems Support</td>
<td>0</td>
</tr>
<tr>
<td>Problems passed to Helpdesk</td>
<td>0</td>
</tr>
<tr>
<td>Problems unresolved</td>
<td>0</td>
</tr>
<tr>
<td>Problems to be re-reported</td>
<td>1</td>
</tr>
</tbody>
</table>
System Support

Context

On Wednesday 22nd of October 1997, I spent two hours, from 1000h to 1200h, observing the activities that Philip Naylor, a member of the System Support team, performed. At the time Philip was not only a member of Systems Support, but also the acting Principal Computing Officer.

Philip works in a small narrow office which he shares with Sam Hollyer (PC Support) and Steve Allman (Networks Support). It struck me as quite a noisy environment, with three 'phones, all of which rang at least once while I was present. It also struck me that Philip was still being interrupted in his work quite frequently.

Observation
APPENDIX N. OBSERVING THE COTS USER-PROBLEM MANAGEMENT SYSTEM

1000 Phil is in an unplanned meeting with the Dean.

1010 Phil arrives back at his office. 'Phone rings. PN answers 'phone. Peter Martin (staff) asks to see PN. PN tells PM to come up immediately. Later I ask PN why he had decided to see PM personally, and why he had decided to see him immediately, rather than, say, scheduling him for some later date/time. PN told me that he decided to handle PM’s problem himself as he (PN) was the expert in the problem area (HTML). He decided to handle it there and then because he judged it to be a short term task and he liked to tackle short-term tasks as quickly as possible. In addition he had not at that time started any other task.

1012 Peter Martin arrives. PM explains problem he has with HTML script. PN tries script.

1015 PN still dealing with PM’s problem.

1017 Heidi Kimber pops her head around the office door. HK sees PN is busy. HK withdraws.

1020 PM is still asking for information and advice about this HTML script that he is writing.

1021 PN finishes solving PM’s problem. PM leaves PN’s office. I ask PN to describe his problem-management system. PN says that there are three levels of problem: first, long-term problems, which he captures on post-it notes, which he sticks to the side of his terminal - such problems last longer than a week; second, there are medium-term problems, which are often described in emails sent by their owners - such problems are usually resolved within a week; and third, there are short-term problems, which are communicated to PN over the 'phone or verbally by people entering his office. Currently PN has the following 10 long-term problems: 1. Looking at ASSET; 2. Re-organising ADA disc; 3. Evaluating Tripwise: alternative security system; 4. Product licencing issues; 5. Setting up a virtual server; 6. Setting up a fax server on the machine Sister; 7. Web Server actions; 8. Monitoring system performance; 9. Maintaining the broken machines in the labs.; 10. Looking at network problems; In answer to one of my questions PN replies that there are no strategic long term problems at this time of year; these are more frequent in the holiday periods, especially the summer holiday.

1023 Julia Dawson (COTS staff; joint acting Principal Computing Officer) comes into the office. She explains that she is scheduled soon to attend a FCC meeting. She asks PN for status of PC problem and for status of other important problems. PN begins reporting status to JD.

1026 PN is still reporting status.
APPENDIX N. OBSERVING THE COTS USER-PROBLEM MANAGEMENT SYSTEM

1034 PN finishes reporting status. JD leaves the office. Stephen Mullen (User Support, reports to PN) asks PN a question about restores. PN gives some information to SM. PN sends SM to JD to obtain more information. PN gives more information to SM.

1036 SM leaves.

1045 PN starts looking through the security reports held as emails (a daily task).

1046 HK reports problem with Netscape. HK asks question. HK leaves. PN explains Netscape problem to me.

1050 PN returns to the security reports.

1054 PN finishes looking at the last security report. PN reads recent (medium-term) emailed problem. PN replies to the problem-owner requesting more information. PN deletes problem from list (deletes the email); it will be re-instated if the problem owner replies.

1056 PN asks Steve Allman (SA) (Member of System Support about setting up a Multia in the machine room. (PN asks this because SA was available and PN had been thinking about it.) I ask PN for his opinion on the usefulness of a computer-based problem management system (CBPMS). PN replied that a CBPMS which helped maintain priorities especially for long-term tasks would be useful. Also some (automated) system of raising priorities as time goes by would be useful.

1104 HK interrupts with more questions on Netscape. PN answers these questions.

1107 HK leaves

1107 PN makes tea and coffee.

1118 Tea and coffee made.

1120 PN tidies desk.

1125 PN updates his logbook.

1125 Ian (staff) inquires about statistics facilities, e.g. SPSS, for Unix. PN sends him to HK and Sam. Ian volunteers information about Solaris.

1128 SM brings in a student with a firewall query. PN sends them to AC. PN is on leave in the afternoon, so he reviews his medium-term problem list for problems that he thinks will take between half and one hour to resolve. He indicates to me that if it seems to be quiet he may start a long-term problem.

1130 PN selects a mailing-list problem. PN begins to work on the mailing-list problem.

1133 Paul Raynor (staff) asks PN for the status of the PC problem. PN replies. PN returns to the mailing-list problem.
1137 Pronob Sarkar (staff) has an HTML query. PN gives him some advice. PS asks about feasibility. PN returns to the mailing-list problem.

1139 PN answers a ’phone call from Tony Solomonides (staff). PN gives TS technical advice over the ’phone: ”Press ”stop” and ”A” key”, etc.

1142 Back to problem.

1145 PN completes mailing-list problem. PN starts to e-mail the owner of the mailing-list problem with instructions on how to test the solution.

1147 PN answers ’phone call from TS, who is trying to boot his machine. PN gives TS advice over the ’phone.

1150 Liz Davis (Helpdesk Manager) enters with disc labels to show to PN. PN is still e-mailing the owner of the mailing-list problem.

1153 PN needs to look up the syntax of something for the instructions. This is held in the adjacent office.

1154 PN typing in instructions to mailing list problem owner.

1156 PN typing in instructions to mailing list problem owner.

1158 PN typing in instructions to mailing list problem owner.

1158 PN refers to syntax.

1159 PN typing in instructions to mailing list problem owner.

1201 PN typing in instructions to mailing list problem owner.

1201 PN searches manual for help on syntax.

1203 I asked PN how he would feel about writing up problem details in a CBPMS. PN replied that, presumably, a CBPMS would capture for re-use instructions e-mailed to problem owners.

1204 e-mail completed and sent.
Analysis

As the transcript indicates, PN recognises three kinds of problems. First, there are long-term problems. PN identifies these as those he thinks will take more than a week to resolve. The details of each long-term problem are recorded on a post-it note and stuck to the side of his work-station. Second, there are medium-term problems. These are ones he thinks that he can resolve within a week. Many of these are transmitted to PN in e-mails. Finally, there are short-term problems. These are the ones that PN thinks that he can fix in a matter of minutes or hours, at the most. Generally these are communicated either over the ’phone or in person. PN likes to tackle short-term problems as quickly as possible, so he usually tackles them immediately.

When PN can select a problem from his lists, he uses the same tactic as SM and selects one that he estimates he can complete in the time currently available.

Like SM, PN was also frequently interrupted in his work, mainly either by User Support colleagues with their own or students’ problems, or by “illegal” direct approaches from students or staff.

The two tables (Table N.4 and Table N.5) presented below summarise the two hour period of observation. They show the problems that were brought from different sources, and the outcome of each attempt to solve a problem. It can be seen that PN engaged with 12 problems during this period. He resolved 10 and passed on 2. Most of the problems were brought to him either by colleagues or by academic staff. However, 2 problems were selected from his medium-term problem list.

During this time, PN engaged in other activities besides tackling problems. For example, he also reported status, reviewed security reports, and initiated the setting up of a Multia machine.

CBPMS requirements

PN indicated that a useful CBPMS would be one that supported a prioritisation of problems, especially long-term problems. A useful CBPMS would also automatically raise a problem’s priority periodically.

When asked about the possibility of having to record problem details in a CBPMS, PN did not raise any objections.

PN expects a CBPMS to support the capture and re-use of problem solutions.

Thus, I inferred the following requirements for a CBPMS from this observation:

1. The CBPMS should provide support for prioritising problems.
2. The CBPMS should provide support for raising the priority of problems periodically and automatically.
3. The CBPMS should provide support for the capture and reuse of solutions to problems.
4. The CBPMS should provide support for recording problem details.
Table N.4: Source of problems encountered in the observation period and their resolution

<table>
<thead>
<tr>
<th>Problem identifier</th>
<th>From problem list</th>
<th>Staff interrupts</th>
<th>Student interrupts</th>
<th>Added to problem list</th>
<th>Resolved</th>
<th>Told to return</th>
</tr>
</thead>
<tbody>
<tr>
<td>html</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>information</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>netscape</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>anon</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>netscape</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>statistics</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>firewall</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>mailing</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>html</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>anon</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>boot</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>2</strong></td>
<td><strong>9</strong></td>
<td><strong>1</strong></td>
<td><strong>0</strong></td>
<td><strong>10</strong></td>
<td><strong>2</strong></td>
</tr>
</tbody>
</table>

Table N.5: Summary of problems encountered in the observation period

<table>
<thead>
<tr>
<th>Information type</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problems</td>
<td>12</td>
</tr>
<tr>
<td>Problems selected from long-term problem list</td>
<td>2</td>
</tr>
<tr>
<td>Problems resolved by Systems Support</td>
<td>10</td>
</tr>
<tr>
<td>Problems passed on</td>
<td>2</td>
</tr>
<tr>
<td>Problems unresolved</td>
<td>0</td>
</tr>
<tr>
<td>Problems to be re-reported</td>
<td>1</td>
</tr>
</tbody>
</table>
Appendix O

The flow of problems into and around the COTS domain
## Pre-printed forms

The form displayed below was used by the COTS personnel to record instances of problems. One form would be completed for each problem instance.

<table>
<thead>
<tr>
<th>NAME:</th>
<th>DATE:</th>
<th>SUBSYSTEM NAME:</th>
</tr>
</thead>
</table>

**HOW DID THE USER COME TO YOU? (TICK ONE OPTION):**

- Direct
- Unix
- PC
- Oracle
- Small
- Network
- NT
- AudioVisual
- COTS
- ITS Helpdesk
- Other

**TRANSACTION TYPE (TICK ONE OPTION):**

- Information request
- Advice request
- Problem report
- Status request
- Borrow
- Collect printout
- Buy article
- Other

**TRANSACTION MEDIUM (TICK ONE OPTION):**

- Face-to-face
- Telephone
- E-mail
- Mail
- Memo
- Other
APPENDIX O. THE FLOW OF PROBLEMS INTO AND AROUND THE COTS DOMAIN

USER TYPE (TICK ONE OPTION)

Student
Academic
Administrator
Other CSM personnel
Non-CSM personnel

---------------------------------------------------------------

WAS THE TRANSACTION FULLY RESOLVED ON THE SPOT?

Yes
No

---------------------------------------------------------------

IF THE ANSWER TO THE PREVIOUS QUESTION WAS "NO", WHO DID YOU PASS THE USER TO?

Direct
Unix
PC
Oracle
Small
Network
NT
AudioVisual
COTS
ITS Helpdesk
Other

---------------------------------------------------------------

ADDED TO LONG_TERM PROBLEM LIST

Yes
No

---------------------------------------------------------------

ADDITIONAL NOTES (OPTIONAL):

360
Results

The following five tables show the problem flow in the COTS domain for each day of a five-day period. (A "*" in a cell means that no data was collected.)

Table O.1: Flow of problems into and around COTS: Monday

<table>
<thead>
<tr>
<th>Name of subsystem</th>
<th>Number of problems presented</th>
<th>Number solved on-the-spot</th>
<th>Number passed on</th>
<th>Number placed on long-term problem list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unix-1</td>
<td>15</td>
<td>11</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Unix-2</td>
<td>21</td>
<td>13</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>PC1</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PC-2</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>NT</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Oracle</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Network</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Atari</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Audio visual</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Small Systems</td>
<td>7</td>
<td>6</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Helpdesk</td>
<td>103</td>
<td>87</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>161</td>
<td>129</td>
<td>26</td>
<td>6</td>
</tr>
</tbody>
</table>
Table O.2: Flow of problems into and around COTS: Tuesday

<table>
<thead>
<tr>
<th>Name of subsystem</th>
<th>Number of problems presented</th>
<th>Number solved on-the-spot</th>
<th>Number passed on</th>
<th>Number placed on long-term problem list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unix-1</td>
<td>30</td>
<td>20</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Unix-2</td>
<td>10</td>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>PC1</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PC-2</td>
<td>9</td>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NT</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Oracle</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Network</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Atari</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Audio visual</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Small Systems</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Helpdesk</td>
<td>125</td>
<td>115</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>198</strong></td>
<td><strong>169</strong></td>
<td><strong>15</strong></td>
<td><strong>14</strong></td>
</tr>
</tbody>
</table>

Table O.3: Flow of problems into and around COTS: Wednesday

<table>
<thead>
<tr>
<th>Name of subsystem</th>
<th>Number of problems presented</th>
<th>Number solved on-the-spot</th>
<th>Number passed on</th>
<th>Number placed on long-term problem list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unix-1</td>
<td>28</td>
<td>14</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>Unix-2</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PC1</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>PC-2</td>
<td>9</td>
<td>6</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>NT</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Oracle</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Network</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Atari</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Audio visual</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Small Systems</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Helpdesk</td>
<td>39</td>
<td>32</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>112</strong></td>
<td><strong>84</strong></td>
<td><strong>22</strong></td>
<td><strong>6</strong></td>
</tr>
</tbody>
</table>
APPENDIX O. THE FLOW OF PROBLEMS INTO AND AROUND THE COTS DOMAIN

Table O.4: Flow of problems into and around COTS: Thursday

<table>
<thead>
<tr>
<th>Name of subsystem</th>
<th>Number of problems presented</th>
<th>Number solved on-the-spot</th>
<th>Number passed on</th>
<th>Number placed on long-term problem list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unix-1</td>
<td>27</td>
<td>16</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>Unix-2</td>
<td>32</td>
<td>15</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>PC1</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>PC-2</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>NT</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Oracle</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Network</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Atari</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Audio visual</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Small Systems</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Helpdesk</td>
<td>91</td>
<td>82</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>157</strong></td>
<td><strong>119</strong></td>
<td><strong>27</strong></td>
<td><strong>11</strong></td>
</tr>
</tbody>
</table>

Table O.5: Flow of problems into and around COTS: Friday

<table>
<thead>
<tr>
<th>Name of subsystem</th>
<th>Number of problems presented</th>
<th>Number solved on-the-spot</th>
<th>Number passed on</th>
<th>Number placed on long-term problem list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unix-1</td>
<td>18</td>
<td>13</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Unix-2</td>
<td>9</td>
<td>7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PC1</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>PC-2</td>
<td>12</td>
<td>9</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>NT</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Oracle</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Network</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Atari</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Audio visual</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Small Systems</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Helpdesk</td>
<td>75</td>
<td>71</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>135</strong></td>
<td><strong>120</strong></td>
<td><strong>12</strong></td>
<td><strong>3</strong></td>
</tr>
</tbody>
</table>
Appendix P

Client’s view of problems, goals, and constraints for new served and serving systems
Introduction

This appendix presents in three tables (a Client’s Problems Table, a Client’s Goals Table, and a Client’s Constraints Table) the client’s view of problems, goals and constraints associated with user-problem management.

The three sections that follow the tables present this data as it was originally captured and presented to the client for validation.

Problems, goals and constraints tables for the client

The Client’s Problems Table is presented in table P.1. The Client’s Goals Table is presented in table P.2. The Client’s Constraints Table is presented in table P.3.
Table P.1: Case study Client’s Problems Table

<table>
<thead>
<tr>
<th>Problem identifier</th>
<th>Short problem description</th>
<th>Problem description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1P1</td>
<td>Some problems are lost</td>
<td>There should be zero &quot;lost&quot; user-problems. User-problems should not be allow to just fade away and eventually be forgotten just because there are no current solutions.</td>
</tr>
<tr>
<td>C1P2</td>
<td>Users find it hard to track problems.</td>
<td>Currently most user-problems are communicated to COTS by word of mouth. They are passed around COTS from one member of the team to another in the same way. Consequently, some time after reporting a problem, it is often difficult for a user to find out the current status of that problem. The chain of COTS staff who have been involved with that problem has to be uncovered in order to find the person currently dealing with the problem and thus ascertain its status.</td>
</tr>
<tr>
<td>C1P3</td>
<td>Problem frequency data is not maintained.</td>
<td>It is not possible currently to determine the frequency of occurrence of particular types of problems. And in general it’s not possible to look for any kinds of patterns in the occurrence of problems.</td>
</tr>
<tr>
<td>C1P4</td>
<td>Hard to access solutions to infrequently occurring but hard-to-fix problems.</td>
<td>It is often difficult to retrieve the details of solutions to complex, difficult, infrequently occurring user-problems each time that they are needed. Searching for the solution usually entails a COTS member of staff trying to remember the date when the problem last occurred and then searching through paper-based files in which the details of the solution might have been recorded.</td>
</tr>
<tr>
<td>C1P5</td>
<td>Prioritising problems is problematic.</td>
<td>Computer Officers are often unsure as to how to prioritise a particular user-problem. They each have certain personal criteria for making such decisions, e.g. give top priority to the user-problem that affects most people. And to a certain extent these sets of criteria overlap. However, sometimes they allow their usual criteria to be overridden by other criteria.</td>
</tr>
<tr>
<td>C1P6</td>
<td>Users receive insufficient problem status information.</td>
<td>Currently, COTS are not good at feeding back the current status of user-problems to the problem-owners.</td>
</tr>
<tr>
<td>C1P7</td>
<td>Managing problems is difficult as they have low visibility.</td>
<td>It is difficult to manage staff who are attempting to resolve user-problems when there is usually no explicit record of each user-problem and its current status to refer to. For example it is currently difficult for the COTS manager to find out which problems are currently being worked on, which problems are blocked and why, which problems are being overlooked, and so on.</td>
</tr>
<tr>
<td>C1P8</td>
<td>Problems may remain unsolved indefinitely.</td>
<td>A user-problem with a low-priority may remain unsolved indefinitely.</td>
</tr>
</tbody>
</table>
Table P.2: Case study Client’s Goals Table

<table>
<thead>
<tr>
<th>Goal identifier description</th>
<th>Short goal description</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1G1</td>
<td>Database of common problem-solution descriptions</td>
<td>Maintain a database of common problem-solution descriptions and provide easy and efficient access for users to this information. Such a system could be an online system.</td>
</tr>
<tr>
<td>C1G2</td>
<td>Uniform user-problem management</td>
<td>Ensure that user-problems are managed in a uniform way irrespective of their system of origin—PC, MAC, Unix, etc.</td>
</tr>
<tr>
<td>C1G3</td>
<td>One rule set</td>
<td>In the computer-based component of the user-problem management system, use only one set of rules, protocols, forms, etc. for managing user-problems.</td>
</tr>
<tr>
<td>C1G4</td>
<td>Automatic priority raising</td>
<td>The priority of each user-problem recorded in the user-problems management system should be automatically raised each time a pre-specified time-period elapses.</td>
</tr>
<tr>
<td>C1G5</td>
<td>Manager monitoring</td>
<td>It must be possible for a manager to use the user-problem management system to see at any time who is working on what, and the status of all current user-problems.</td>
</tr>
<tr>
<td>C1G6</td>
<td>Regular status reports</td>
<td>Each user should be informed of the status of his or her user-problem at sensible times during its lifecycle.</td>
</tr>
<tr>
<td>C1G7</td>
<td>Inadvertently no goal was designated as G7.</td>
<td>Inadvertently no goal was designated as G7.</td>
</tr>
<tr>
<td>C1G8</td>
<td>Log faults from anywhere</td>
<td>It should be possible for users to log faults with the COTS team from a variety of geographical locations.</td>
</tr>
<tr>
<td>C1G9</td>
<td>Match problem to expert</td>
<td>COTS should provide a mechanism whereby each user-problem obtains the level of expertise it requires for its resolution as quickly as possible.</td>
</tr>
<tr>
<td>C1G10</td>
<td>Single problem owner</td>
<td>For the life of a problem it should be “owned” by one COTS team member.</td>
</tr>
<tr>
<td>C1G11</td>
<td>Support in one room</td>
<td>The User Support Team should be housed in the same room.</td>
</tr>
<tr>
<td>C1G12</td>
<td>Multi-skilled support</td>
<td>User Support team members should become increasingly able to handle problems from a variety of computer system sources—PC, Unix, MAC, etc.</td>
</tr>
<tr>
<td>C1G13</td>
<td>Multi-knowledge</td>
<td>User Support team members should become increasingly aware of each others’ areas of expertise and skills.</td>
</tr>
<tr>
<td>C1G14</td>
<td>Very happy users</td>
<td>The level of users’ happiness with the service provided by COTS should be increased until it approaches all users being “very happy”.</td>
</tr>
<tr>
<td>C1G15</td>
<td>Maintaining problem record</td>
<td>It must be possible to add details to any record of a problem throughout its lifecycle.</td>
</tr>
</tbody>
</table>
Table P.3: Case study Client’s Constraints Table

<table>
<thead>
<tr>
<th>Constraint identifier</th>
<th>Short constraint description</th>
<th>Constraint description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1C1</td>
<td>Timescales-1</td>
<td>The bulk of the requirements should be completed by September 30th, 1998.</td>
</tr>
<tr>
<td>C1C2</td>
<td>Timescales-2</td>
<td>A complete system should be up and running by September the 30th, 1999.</td>
</tr>
<tr>
<td>C1C3</td>
<td>Budget</td>
<td>Development costs should not exceed £6,000, i.e. the cost of one placement student for one year.</td>
</tr>
<tr>
<td>C1C4</td>
<td>Staff</td>
<td>The introduction of any new system cannot depend on any increase in the number of COTS team members, nor must it cause any decrease in the numbers of COTS team members.</td>
</tr>
<tr>
<td>C1C5</td>
<td>Development methods</td>
<td>The choice of development method is not constrained.</td>
</tr>
<tr>
<td>C1C6</td>
<td>Development languages</td>
<td>The language used should be one for which there is currently expertise within COTS since the COTS team will probably maintain the system. So the candidates are the following: Perl, C, or HTML.</td>
</tr>
<tr>
<td>C1C7</td>
<td>Multi-platform integrated architecture</td>
<td>The system should work in a uniform way on a variety of system platforms—Unix, PC, MAC, and so on—and be integrated in the sense that a global view of all user-problems can be accessed and updated from any platform.</td>
</tr>
<tr>
<td>C1C8</td>
<td>Performance</td>
<td>The users of the system must be happy with the time and effort that is required to log problems.</td>
</tr>
</tbody>
</table>
Client’s view of problems associated with user-problem management:

On Friday the 18th of October, 1997, I interviewed the client, Jon Ward, and asked him questions about his view of problems, goals, and constraints associated with managing user-problems. This report contains his view of the problems.

QUESTION: Can you describe some of the problems that you would like to see eliminated or attenuated in any new computer-based problem management system.

1. Lost user-problems:

There should be zero ‘lost’ user-problems. User-problems should not be allowed to just fade away and eventually be forgotten just because there are no current solutions.

2. Traceability:

Currently most user-problems are communicated to COTS by word of mouth. They are passed around COTS from one member of the team to another in the same way. Consequently, some time after reporting a problem, it is often difficult for a user to find out the current status of that problem. The chain of COTS staff who have been involved with that problem has to be uncovered in order to find the person currently dealing with the problem and thus ascertain its status. This has to be done by asking questions. If one or more people in the chain are not around then it is not possible to complete the construction of the chain.

3. Rate of occurrence unavailable for each type of user-problem:

It is not possible currently to determine the frequency of occurrence of particular types of problems. And in general it’s not possible to look for any kinds of patterns in the occurrence of problems. This is because problem details are not systematically recorded and saved. If this were to be done then it would be possible to search for and identify problem patterns and then be in a position to take more effective measures in dealing with those problems. For example, for a recurring problem, perhaps a helpsheet could be written and given to each user reporting an instance of that problem. This would save time for COTS staff, i.e. the time spent resolving the problem each time it is reported.
4. Solutions unavailable for infrequently occurring but hard-to-fix user-problems:

It is often difficult to retrieve the details of solutions to complex, difficult, infrequently occurring user-problems each time that they are needed. Searching for the solution usually entails a COTS member of staff trying to remember the date when the problem last occurred and then searching through paper-based files in which the details of the solution might have been recorded.

5. Prioritising user-problems:

Computer Officers are often unsure as to how to prioritise a particular user-problem. They each have certain personal criteria for making such decisions, e.g. give top priority to the user-problem that affects most people. And to a certain extent these sets of criteria overlap. However, sometimes they allow their usual criteria to be overridden by other criteria. For example, a member of staff can raise the priority of his or her own user-problem just by being domineering; in such cases the Computer Officers may deal quickly with the user-problem in order to get rid of the problem-owner.

6. Little or no feedback to problem-owners on problem-status:

Currently, COTS are not good at feeding back the current status of user-problems to the problem-owners.

(I suggested that they should be informed of the status of their problem at each significant event in the user-problem’s life-history.)

7. Low visibility of user-problems makes managing difficult:

It is difficult to manage staff who are attempting to resolve user-problems when there is usually no explicit record of each user-problem and its current status to refer to. For example it is currently difficult for the COTS manager to find out which problems are currently being worked on, which problems are blocked and why, which problems are being overlooked, and so on.

8. Problems may remain unsolved indefinitely

A user-problem with a low-priority may remain unsolved indefinitely.
Client’s view of goals associated with user-problem management:

On Friday the 18th of October, 1997, I interviewed the client, Jon Ward, and asked him questions about his view of problems, goals, and constraints associated with managing user-problems. This report contains his view of the goals.

For some of the goals stated by the client, I have extrapolated the goal to generate one or more higher-level goals (marked SGn) to the satisfaction of which, it seems, it contributes. These higher-level goals can be viewed as providing a rationale for the lower-level ones to which they are linked.

For the client’s goal related to levels of service provision (goal G9), an outline method of satisfying the goal was also suggested. This is include here in the asterisk box.

QUESTION: Can you outline some of the goals that you would like to see satisfied or satisficed by a new user-problem management system.

G1. Maintain a database of common problem-solution descriptions and provide easy and efficient access for users to this information. Such a system could be an online system.

SG1: Reduce the number of visits made by users to the Helpdesk or Computer Officers’ offices.

(Goal G1 supports goal SG1)

SG2: Providing more time for COTS staff to spend on solving user-problems.

(Goal SG1 supports goal SG2)

G2. Ensure that user-problems are managed in a uniform way irrespective of their system of origination - PC, MAC, Unix, etc.

G3. In the computer-based component of the user-problem management system use only one set of rules, protocols, forms,
etc. for managing user-problems.

(Goal G3 support goal G2)

SG3: Make it easy for users to learn to use the user-problem management system.

(Goal G2 support goal SG3)

SG4: Make it easy for COTS staff to work on user-problems from a variety of system sources.

(Goal G2 support goal SG4)

G4. The priority of each user-problem recorded in the user-problem management system should be automatically raised each time a pre-specified time period elapses.

SG5: Ensure that eventually within a sensible period of time every problem will be tackled.

(Goal G4 supports goal SG5)

G5. It must be possible for a manager to use the user-problem management system to see at any time who is working on what and the status of all current user-problems.

SG6: Manager can act more effectively in managing user-problems.

(Goal G5 supports goal SG6)

G6. User should be informed of the status of his/her problem at sensible times during its lifecycle.

SG1: Reduce the number of visits made by users to the Helpdesk or Computer Officers’ offices.

(Goal G6 supports goal SG1)

SG8: Maximise users’ satisfaction with service provided by COTS.

(Goal G6 supports goal SG8)
G8. It should be possible for users to log faults with the COTS team from a variety of geographical locations, e.g. from the laboratories, from staff offices, from outside the faculty, and from outside the University.

G9. COTS should provide a mechanism whereby each user-problem obtains the level of expertise it requires for its resolution as quickly as possible.

***************************************************************
* One way for achieving this is set out below. This is the way * things are currently done in COTS.
* All student user-problems should be brought to the Helpdesk * initially. If the Helpdesk cannot solve them, Helpdesk staff * arrange for the student to see a member of User Support. If * User Support cannot help, they pass the problem over to * Systems Support.
* A design has been proposed in which a computer-based system * supports this organisational structure. This design is held * in the document clients_sociotechnical_system.
***************************************************************

G10. For the life of a problem it is "owned" by only one COTS team member.

This makes it very easy for a user to find out the status of his/her problem since he/she will always be dealing with the same person from COTS. (However, since in practice user-problems are often passed from Computer Officer to Computer Officer, there would need to be a second concept of ownership for COTS use.)

SG9. Make it as easy as possible for users to track their user-problems.

(Goal G10 supports goal SG9)

G11. The User Support team should be housed in the same room.

G12. User Support team members should each become increasingly more able to handle problems from a variety of computer system
sources: PC, Unix, MACS, etc.

G13. User Support team members should become increasingly aware of each others’ areas of expertise and skills.

(Goal G11 supports goals G12 and G13)

G14. Although most users are currently "happy" with the service provided by COTS, the level of happiness should be increased until it approaches all users being "very happy".

G15. It must be possible to add details to any record of a problem throughout its lifecycle.

SG10. It must be possible to record, save, and amend the details of problems reported by users. And it must be possible to do this quickly and easily.
Client’s view of constraints associated with user-problem management:

On Friday the 18th of October, 1997, I interviewed the client, Jon Ward, and asked him questions about his view of problems, goals, and constraints associated with managing user-problems. This report contains his view of the constraints.

QUESTION: Do you have specific constraints on the development of a computer-based user-problem management system?

* Timescales:

  o The bulk of the requirements phase (notwithstanding expected iterations back to this phase during later phases) should be completed by the end of the current academic year, i.e. by September 30th 1998.

  o A complete system should be up and running by the end of the following academic year, i.e. by September 30th 1999.

* Budget:

  o 5-6K pounds, i.e. one placement student (or equivalent) for one year.

* Staff:

  o The introduction of any new system cannot depend on an increase in the number of COTS team members, nor must it cause any decrease in the number of COTS team members.

* Development methods:

  o No preferences.

* Development languages:

  o Since the COTS team will probably maintain the system, it is felt that the language used should be one for which there is currently expertise within COTS. So the candidates are the following:

    - Perl
    - C
    - HTML
* Multi-platform integrated system architecture:

- The system should work in a uniform way on a variety of system platforms - Unix, PC, MAC, etc. - and be integrated in the sense that a global view of all user-problems can be accessed and updated from any platform.

* Performance:

- The users of the system must be happy with the time and effort that is required to log problems.

  This implies that it will easy for users to enter information quickly. This in turn implies that the user interface will help to minimise typing. So user-interface features might include, for example, click buttons, and lists of problem areas to choose from a menu.
Appendix Q

Other stakeholders’ views of problems, goals, and constraints for new served and serving systems
APPENDIX Q. OTHER STAKEHOLDERS’ VIEWS OF GOALS FOR NEW SERVED SYSTEMS

Introduction

The information presented in the three sections of this appendix was derived during the step “Review other stakeholders’ views of problems, goals, and constraints”. It is described in chapter 7.

Other stakeholders’ views of problems associated with user-problem management:

In this section, the views of each of the systems experts and the Helpdesk manager are presented on problems associated with the way COTS manage long-term user-problems.

COTS1:

1. Lost problems:

   COTS1 thinks that some user-problems may be lost.

2. Reactive system:

   COTS1’s system is too reactive in the sense that COTS1 often has to react in real-time to users bringing their problems.

3. Sensitive system

   COTS1’s system is too sensitive to social pressure applied by individuals e.g. some members of staff.

4. Unresolvable problems:

   COTS1 does not think such problems exist.

COTS2:

1. Hard to say no:

   COTS2 finds it difficult to say "no" to users, i.e. to turn them away or make them wait when they have problems.

2. Lack of explicit problem stack:

   COTS2 see the lack of an explicit stack of user-problems as a problem, because:

   - it leads to some user-problems being forgotten
   - it leads to a low or zero ability to prioritise and
schedule problems.

3. Unresolvable problems:

COTS2 indicated that such problems are placed on the back-burner. These problems may be brought back to life by opportunistic discoveries or events, e.g. finding the source of a required package on the net.

COTS3:

1. Not integrated:

   The three different systems used by COTS3 are not integrated in any way.

2. Lack of user-problem visibility:

   COTS3 cannot see what other Computer Officers are doing who working on related tasks.

3. Lack of user-problem record:

   COTS3 cannot easily tell whether a given user-problem has ever been tackled previously.

4. NT: slow

   The purpose-built spread-sheet is slow to load up.

5. NT: time-consuming to page through

   The purpose-built spread-sheet has to be paged through one page at a time in order to access a particular problem.

6. ToDoList: Insufficient information storable:

   The ToDoList does not enable enough information about a user-problem to be stored.

7. ToDoList: Lacks prompts

   The ToDoList does not enable specific categories of information to be prompted for.

8. Notebook: Unformatted

9. Notebook: Lose pages
Sometimes loose pages are lost from the notebook.

10. Unresolvable problems (major):

In the case of unresolvable major problems COTS3 contacts the appropriate technical support team.

11. Unresolvable problems (minor):

Unresolvable minor problems go to the bottom of the list of problems awaiting resolution (usually after some fix or work-around has been installed).

Sometimes opportunistic discovery of relevant information may trigger re-opening of a particular user-problem.

COTS4:

1. Uncertain prioritisation:

Prioritisation is ad hoc; one can never know if right decision was made.

2. Lost problems:

Some problems "drop off the list".

3. Lack of statistics:

No statistics are kept concerning user-problems.

4. Lack of information on time taken to resolve user-problems:

5. Forget information:

Sometimes COTS4 overlooks or forgets to record important information.

6. Unresolvable problems:

Generally these problems remain on the back-burner. Sometimes they solve themselves by going away. For example member of staff has an unresolvable problem, but then leaves the University.

COTS5:
1. Problem tracking and recording is difficult:

Tracking the status of a user-problem can be difficult, particularly if is necessary to go outside the faculty for help.

2. Tracking the status of a problem is particularly difficult when it has become the responsibility of more than one team within COTS.

3. It is not possible to produce accurate statistics showing what work has been accomplished. So for example, it is difficult to find out how much time has been wasted on non-CSM users.

COTS6:

1. Unresolvable problems:

These are placed on the back-burner where they often tend to fade away. Some are given to other Computer officers.

2. Interrupt driven:

Users may interrupt COTS6 at any time with new problems.

3. Never started:

Some user-problems which require long-term tasks for their resolution are never started.

COTS7:

1. Rudeness from users.

2. Male chauvinism:

Some male users do not want to communicate their problems to COTS7 since COTS7 is female.

Users’ view

In this section some users’ views are presented of problems associated with the way COTS manage long-term user-problems.

1. Some problems reported by users are ignored and/or
forgotten.
2. Managers’ problems are not always given a high enough priority.

Other stakeholders’ views of goals associated with user-problem management:

This section describes from three points of view—the systems expert’s, the Helpdesk manager’s, and the users’—goals that they would like to see satisfied in any future system for managing long-term user-problems.

COTS1:

1. Stack of user-problems:

   COTS1 would like to work from an explicit stack of problems which have have either been scheduled for COTS1 or scheduled by COTS1.

2. Minimise user contact:

   COTS1 would like less direct involvement with users.

3. Problems via Helpdesk:

   COTS1 would like to receive all user-problems via the Helpdesk.

4. Cut off when necessary:

   COTS1 would like to be able to completely terminate contact with users when necessary.

5. Archive of user-problems

   COTS1 would find it useful to have a key-word searchable archive of problems since:
   
   - it would help to refresh COTS1’s memory of solutions
   - it could be used for sharing knowledge with other staff and new staff

COTS2:

1. Control of problem reception:
COTS2 would like to control the way that user-problems are received, in order to avoid being bombarded with problems. For example, if all user-problems were to be emailed to COTS2 then they could be tackled one at a time.

2. COTS2 would like an assistant - human or computer-based - to prioritise and schedule user-problems for COTS2. However, COTS2 would also like to be able to reschedule these tasks sometimes so that, e.g. COTS2 could select an interesting user-problem to tackle when he wanted to.

3. COTS2 would like to be able to make users aware of when he is free to deal with their user-problems.

4. COTS2 would like to keep a permanent log for two reasons:
   1. It would be personally satisfying to review what has been achieved.
   2. The log might serve as the base of reusable knowledge from telephone numbers and addresses to problem solutions.

COTS3:

1. Integrated, global system:

   Any user-problem management system should be integrated and global in the sense that it would support user-problems from all supported computer systems: MACs, PCs, Unix, etc.

2. Easy access to useful services:

   On a user-problem management system it should be easy to obtain the following services:
   - access information on user-problems
   - do searches on user-problems
   - make ad hoc queries about user-problems.

3. High performance:

   A user-problem management system should be high performing.

4. Automatic reminders:

   A user-problem management system should automatically remind the Computer Officers of flagged tasks.
5. Easy to schedule and re-schedule user-problems

Each Computer Officer should be able to schedule and re-schedule lists of user-problems.

6. Users email/’phone in their problems

Users should be encouraged to email and ’phone in problems rather than bringing them in personally.

7. Academic staff to visit Helpdesk

Academic staff should be encouraged to bring their user-problem to COTS via the Helpdesk

**COTS4:**

1. More information available online:

   More information should be available "on tap", e.g. Netwire Magazine "frequently asked questions" should be available online.

2. More time for strategic work:

   Firmer boundaries should be drawn around responsibilities at work to give more time for strategic work. More user-support work could be given to junior Computing Officers and less to senior Computing Officers.

3. Good performance:

   Access to a Problem Management System database must take less than 10 seconds.

**COTS5:**

1. X-Platform:

   A User-Problem Management System should be based on an x-platform, e.g. be based on the Web.

2. Quick:

   A computer-based User-Problem Management System needs to be quick to use.
3. Central database/ emailed results:

It should be possible for students and staff to log faults and queries to a central database. Such users should be emailed with the resolution of their queries and fault reports.

4. Statistics:

A computer-based User-Problem Management System should be used to produce statistics on problem-solving, e.g., times taken to solve problems, throughput metrics, number of problems reported in unit time, number of problems resolved in unit time.

5. Book C0s' time.

Users (staff and students) should be able to book times to meet COTS support team members.

6. Students and staff should be made aware of exactly what C0s can and cannot do within given time-scales.

COTS6:

1. Reduce number of user-problems brought to COTS6
2. Senior C0s should only work on long-term tasks.
3. C0s should be allowed to work without interrupts.
4. C0s should receive user-problems in a controlled way, e.g. only at certain times of the day, and then only by email, for example.
5. C0s should have scheduled times for being in interrupt and interrupt-free mode.

COTS7:

1. Not manage consumables

COTS7 would prefer to manage just user-problems at the Helpdesk and not other things like consumables.

2. Employ a second Helpdesk Computer Officer
APPENDIX Q. OTHER STAKEHOLDERS’ VIEWS OF GOALS FOR NEW SERVED SYSTEMS

3. Technical expert on helpdesk

Have one person with wide-ranging technical ability continually present at the Helpdesk

Users’ view:

This section describes the goals that some of the users would like to see satisfied in any future system for managing long-term user-problems.

1. Employ more computer officers

2. Employ more Helpdesk personnel.

3. Enable computer officers to spend more time on problems.

4. Provide more technical training for Helpdesk personnel.

5. Provide customer handling training for Helpdesk personnel.

6. Do not use freshers on the Helpdesk

7. Give priority to academic staff problems.

8. Increase staff coverage for each supported system, e.g. more staff support for MACs

9. Provide Unix and PC support whenever the laboratories are open.

10. Allow users to email problems to Helpdesk personnel and computer officers, i.e. publish relevant email points.

11. Identify a system expert for each system - someone who is prepared to field quite tough problems.

12. Have set times when computer officers would be dedicated to tackling user problems. Publicize them.

13. Publicize skill profiles for each member of the COTS team

14. Log problems to prevent verbal requests from being lost.
Other stakeholders’ views of constraints associated with user-problem management:

This section describes a set of constraints associated with any future computer-based sociotechnical system for managing long-term user-problems. It was gathered by the RE through observing COTS staff carrying out their day to day activities, and asking questions about possibilities for a future computer-based system. As part of the investigation some time was spent observing COTS staff doing their normal day to day work. During these observation sessions questions were asked about the feasibility of various features of a computer-based system for managing long-term user-problems.

Helpdesk

In the following paragraphs K and M are Helpdesk assistants.

K was how feasible it would be for her to type in details of problems if a computer-based problem management system were used on the Helpdesk.

K says that she would find typing in a lot of details to be too time-consuming. And that this would be particularly true when she was busy, for example during the lunch hour.

K says, after it is suggested to her, that she wouldn’t mind pressing one or two keys for each interaction with a user.

She indicates that there might be a problem at shift change-over. The RE assumes that this is because the Help Assistants cannot wait around after a change-over because e.g. they must go to lectures.

K agrees, after it is suggested to her, that typing in a user’s system id, while using a computer-based problem management system might be feasible. The id could be used as a key in order to obtain more complete user details.

M suggests that to type in a handful of key strokes would be feasible. Thus she agrees with K’s opinion.

M suggests that Helpdesk personnel could make notes by hand either during or after a transaction (or both). These could be entered into the system at a later (quieter) time.
Appendix R

Tables of stakeholders’ problems, goals, and constraints
Introduction

This appendix tabulates, in a series of tables, the problems some of the stakeholders in the case study have with the current served and serving systems, and the goals and constraints that they have expressed for new served and serving systems.

The client’s problems, goals, and constraints are tabulated in appendix P (see page 364) in a Client’s Problems Table, Client’s Goals Table, and Client’s Constraints Table.

Each of the other stakeholders problems, goals and constraints are tabulated in a similar set of tables, the raw data for which may be found in appendix Q (page 377).

Stakeholder Stakeholder-one’s Problems Table

Table R.1: Case study Stakeholder Stakeholder-one’s Problems Table

<table>
<thead>
<tr>
<th>Problem identifier</th>
<th>Short problem description</th>
<th>Problem description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1P1</td>
<td>Lost problems.</td>
<td>Stakeholder-one thinks that some user-problems maybe lost.</td>
</tr>
<tr>
<td>S1P2</td>
<td>Users bypass helpdesk.</td>
<td>Stakeholder-one thinks that some users bypass the helpdesk and bring their problems directly to Computer officers.</td>
</tr>
<tr>
<td>S1P3</td>
<td>Social pressure determines problem priority.</td>
<td>Stakeholder-one feels that he is too responsive to social pressure applied by individuals, e.g. some members of staff.</td>
</tr>
</tbody>
</table>
Stakeholder Stakeholder-one’s Goals Table

Table R.2: Case study Stakeholder Stakeholder-one’s Goals Table

<table>
<thead>
<tr>
<th>Goal identifier</th>
<th>Short goal description</th>
<th>Goal description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1G1</td>
<td>Explicit problem stack.</td>
<td>Stakeholder-one would like to work from an explicit stack of problems which have either been scheduled for him or by him.</td>
</tr>
<tr>
<td>S1G2</td>
<td>Minimal user contact.</td>
<td>Stakeholder-one would like less direct involvement with users.</td>
</tr>
<tr>
<td>S1G3</td>
<td>Problems via helpdesk.</td>
<td>Stakeholder-one would like to receive all user-problems via the helpdesk.</td>
</tr>
<tr>
<td>S1G4</td>
<td>Cut off when required.</td>
<td>Stakeholder-one would like to be able to completely prevent contact with users when necessary.</td>
</tr>
<tr>
<td>S1G5</td>
<td>Searchable user-problem archive.</td>
<td>Stakeholder-one would find it useful to have a keyword searchable archive of user problems since it would help to refresh his memory of solutions and it could be used for sharing knowledge with new staff.</td>
</tr>
</tbody>
</table>

Stakeholder Stakeholder-one’s Constraints table

No constraints were elicited from stakeholder-one.
### Stakeholder Stakeholder-three’s Problems Table

Table R.3: Case study Stakeholder Stakeholder-three’s Problems Table

<table>
<thead>
<tr>
<th>Problem identifier</th>
<th>Short problem description</th>
<th>Problem description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3P1</td>
<td>Unintegrated problem management system.</td>
<td>The three different systems used by Stakeholder-three are not integrated.</td>
</tr>
<tr>
<td>S3P2</td>
<td>Poor user-problem visibility.</td>
<td>Stakeholder-three cannot see what other Computer Officers are doing who are working on related tasks.</td>
</tr>
<tr>
<td>S3P3</td>
<td>Lack of user-problem record.</td>
<td>Stakeholder-three cannot easily tell whether a given user-problem has ever been tackled previously.</td>
</tr>
<tr>
<td>S3P4</td>
<td>Spreadsheet slow.</td>
<td>The purpose-built spreadsheet is slow to load up.</td>
</tr>
<tr>
<td>S3P5</td>
<td>Time-consuming to page through.</td>
<td>Stakeholder-three’s purpose-built spreadsheet has to be paged through one page at a time.</td>
</tr>
<tr>
<td>S3P6</td>
<td>To-Do-List poor storage.</td>
<td>Stakeholder-three’s online To-Do-List does not enable enough information about a user-problem to be stored.</td>
</tr>
<tr>
<td>S3P7</td>
<td>To-Do-List lacks prompts.</td>
<td>The To-Do-List does not enable specific categories of information to be prompted for.</td>
</tr>
<tr>
<td>S3P8</td>
<td>Unformatted notebook.</td>
<td>Stakeholder-three’s offline notebook is unformatted.</td>
</tr>
<tr>
<td>S3P9</td>
<td>Notebook loses pages.</td>
<td>Sometimes loose pages are lost from the notebook.</td>
</tr>
<tr>
<td>S3P10</td>
<td>Unresolvable problems (minor)</td>
<td>Minor unresolvable problems go to the bottom of the list of problems awaiting resolution; their resolution may be consequently delayed or put off indefinitely.</td>
</tr>
</tbody>
</table>
## Stakeholder Stakeholder-three’s Goals Table

Table R.4: Case study Stakeholder-three’s Goals Table

<table>
<thead>
<tr>
<th>Goal identifier</th>
<th>Short goal description</th>
<th>Goal description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3G1</td>
<td>Integrated global system.</td>
<td>Any user-problem management system should be integrated and global: it should support user-problems from all supported computer systems; MACs, PCs, Unix, etc.</td>
</tr>
<tr>
<td>S3G2</td>
<td>Easy access to problem information.</td>
<td>In a user-problem management system it should be easy to access problem information.</td>
</tr>
<tr>
<td>S3G3</td>
<td>Easy access to problem search service.</td>
<td>In a user-problem management system it should be easy to access a problem search service.</td>
</tr>
<tr>
<td>S3G4</td>
<td>Easy access to ad hoc queries.</td>
<td>In a user-problem management system it should be easy to access an ad hoc problem query service.</td>
</tr>
<tr>
<td>S3G5</td>
<td>High performance.</td>
<td>A user-problem management system should be high performing.</td>
</tr>
<tr>
<td>S3G6</td>
<td>Automatic task reminder.</td>
<td>A user-problem management system should automatically remind COs of flagged tasks.</td>
</tr>
<tr>
<td>S3G7</td>
<td>Easy to schedule/reschedule problems.</td>
<td>Each Computer officer should be able to schedule and reschedule user-problems.</td>
</tr>
<tr>
<td>S3G8</td>
<td>Discourage personal contact.</td>
<td>Users should be encouraged to e-mail and phone in their problems rather than bringing them in personally.</td>
</tr>
<tr>
<td>S3G9</td>
<td>Academic staff via helpdesk.</td>
<td>Academic staff should be encouraged to bring their user-problems to the Helpdesk in the first instance.</td>
</tr>
</tbody>
</table>

## Stakeholder Stakeholder-three’s Constraints Table

No constraints were elicited from Stakeholder-three.
Stakeholder User-one’s Problems Table
No problems were elicited from User-one.

Stakeholder User-one’s Goals Table
Table R.5: Stakeholder User-One’s Goals Table

<table>
<thead>
<tr>
<th>Goal identifier</th>
<th>Short goal description</th>
<th>Goal description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UG1</td>
<td>More COs.</td>
<td>Employ more COs.</td>
</tr>
<tr>
<td>UG2</td>
<td>More Helpdesk personnel.</td>
<td>Employ more Helpdesk personnel.</td>
</tr>
<tr>
<td>UG3</td>
<td>More time on problems.</td>
<td>Enable computer officers to spend more time on problems.</td>
</tr>
<tr>
<td>UG4</td>
<td>Helpdesk technical training.</td>
<td>Provide more technical training for Helpdesk personnel.</td>
</tr>
<tr>
<td>UG5</td>
<td>Helpdesk interpersonal skills training.</td>
<td>Provide more interpersonal skills training for Helpdesk personnel.</td>
</tr>
<tr>
<td>UG6</td>
<td>No novices on Helpdesk.</td>
<td>Do not use freshers on the Helpdesk.</td>
</tr>
<tr>
<td>UG7</td>
<td>Staff problems - higher priority.</td>
<td>Give priority to academic staff problems.</td>
</tr>
<tr>
<td>UG8</td>
<td>More computer-system support.</td>
<td>Increase staff coverage for each supported system, e.g. Mac, etc.</td>
</tr>
<tr>
<td>UG9</td>
<td>Unix/PC support in open labs.</td>
<td>Provide Unix and PC support whenever the labs are open.</td>
</tr>
<tr>
<td>UG10</td>
<td>Support e-mailing problems.</td>
<td>Allow users to e-mail problems to Helpdesk personnel and COs, so publish e-mail points.</td>
</tr>
<tr>
<td>UG11</td>
<td>Identify experts.</td>
<td>Identify a systems expert for each system - someone who is prepared to field quite tough problems.</td>
</tr>
<tr>
<td>UG12</td>
<td>COs available at set times.</td>
<td>Set and publicise times when COs would be dedicated to tackling user-problems.</td>
</tr>
<tr>
<td>UG13</td>
<td>Publish CO expertise.</td>
<td>Publicise skills profile for each member of the COTS team.</td>
</tr>
<tr>
<td>UG14</td>
<td>Log problems.</td>
<td>Log problems to prevent verbal requests from being lost.</td>
</tr>
</tbody>
</table>

393
Stakeholder User-one’s Constraints Table

No constraints were elicited from User-one.
Appendix S

Transformation of stakeholders’ problems into goals:
Table S.1: Transforming the client’s problem descriptions to corresponding goal descriptions

<table>
<thead>
<tr>
<th>Problem identifier</th>
<th>Short problem description</th>
<th>Goal identifier</th>
<th>Short goal description</th>
<th>Goal description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1P1</td>
<td>Some problems are lost.</td>
<td>C1GP1</td>
<td>No lost user-problems.</td>
<td>In any new system the number of user-problems that become lost should be zero.</td>
</tr>
<tr>
<td>C1P2</td>
<td>Users find it hard to track problems.</td>
<td>C1GP2</td>
<td>Facilitate problem tracking.</td>
<td>In any new system it must be easy for a user to determine which COTS personnel have been involved with resolving their problem and the order of their involvement.</td>
</tr>
<tr>
<td>C1P3</td>
<td>Problem frequency data is not maintained.</td>
<td>C1GP3</td>
<td>Facilitate determination of problem frequency data.</td>
<td>In any new system it must be easy to determine the frequency of occurrence of a particular problem and of particular types of problem.</td>
</tr>
<tr>
<td>C1P4</td>
<td>Hard to access solutions to infrequently occurring but hard-to-fix problems.</td>
<td>C1GP4</td>
<td>Provide access to problem solutions.</td>
<td>In any new system it should be possible to access easily the details of solutions/resolutions of rare but hard-to-fix problems which have occurred before.</td>
</tr>
<tr>
<td>C1P5</td>
<td>Prioritising problems is problematic.</td>
<td>C1GP5</td>
<td>Prioritise user-problems.</td>
<td>In any new system, user-problems should be assigned a priority according to an agreed set of criteria. User-problems should be tackled in priority order. Users should be discouraged from undermining the priority system.</td>
</tr>
<tr>
<td>C1P6</td>
<td>Users receive insufficient problem status information.</td>
<td>C1GP6</td>
<td>Provide problem status to users periodically.</td>
<td>In any new system a problem owner should be informed of the current status of his/her problem at significant checkpoints in its history.</td>
</tr>
<tr>
<td>C1P7</td>
<td>Managing problems is difficult as they have low visibility.</td>
<td>C1GP7</td>
<td>Facilitate access to user-problem information.</td>
<td>In any new system it must be easy to ascertain what problems are in the system, their status, who is currently working on each, who has worked on each, and so on.</td>
</tr>
<tr>
<td>C1P8</td>
<td>Some problems may remain unsolved indefinitely.</td>
<td>C1GP8</td>
<td>Tackle all user-problems within a reasonable time period.</td>
<td>In any new system ensure that all user-problems are tackled within a reasonable time.</td>
</tr>
</tbody>
</table>
Table S.2: Transforming stakeholder one’s problems to goals

<table>
<thead>
<tr>
<th>Problem identifier</th>
<th>Short problem description</th>
<th>Goal identifier</th>
<th>Short goal description</th>
<th>Goal description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1P1</td>
<td>Lost problems.</td>
<td>S1GP1</td>
<td>No lost user-problems.</td>
<td>In any new system the number of user-problems that become lost should be reduced, ideally to zero. Compare with C1GP1.</td>
</tr>
<tr>
<td>S1P2</td>
<td>Users bypass Helpdesk.</td>
<td>S1GP2</td>
<td>Reduce direct user/CO contact.</td>
<td>In any new system the frequency with which users bring their problems to COs should be significantly reduced (e.g. by at least 50%).</td>
</tr>
<tr>
<td>S1P3</td>
<td>Social pressure determines problem priority</td>
<td>S1GP3</td>
<td>Allocate problem priorities rationally</td>
<td>In any new system problems should be prioritised rationally and COs should be supported in resisting social pressure to undermine prioritisation.</td>
</tr>
</tbody>
</table>
Table S.3: Transforming stakeholder three’s problem descriptions to corresponding goal descriptions

<table>
<thead>
<tr>
<th>Problem identifier</th>
<th>Short problem description</th>
<th>Goal identifier</th>
<th>Short goal description</th>
<th>Goal description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3P1</td>
<td>Unintegrated PMS.</td>
<td>S3GP1</td>
<td>Manage all user-problems uniformly.</td>
<td>Manage all user-problems in a uniform way.</td>
</tr>
<tr>
<td>S3P2</td>
<td>Poor user-problem visibility.</td>
<td>S3GP2</td>
<td>Easy to view status of all problems.</td>
<td>In any new system it should be possible to easily “see” status of problems worked on by other COs.</td>
</tr>
<tr>
<td>S3P3</td>
<td>Poor problem-solution recording.</td>
<td>S3GP3</td>
<td>Maintain permanent problem-solution record.</td>
<td>In any new system it should be possible to determine whether a user-problem has ever been previously tackled.</td>
</tr>
<tr>
<td>S3P4</td>
<td>Slow to load personal PMS.</td>
<td>S3GP4</td>
<td>Fast access to problems.</td>
<td>Any new system should be accessible within 10 seconds.</td>
</tr>
<tr>
<td>S3P5</td>
<td>Slow paging of personal PMS.</td>
<td>S3GP5</td>
<td>Fast access to problem details.</td>
<td>In any new system COTS staff must find they can access problem information at an acceptable speed.</td>
</tr>
<tr>
<td>S3P6</td>
<td>Insufficient information in personal PMS.</td>
<td>S3GP6</td>
<td>Store all relevant problem data.</td>
<td>In any new system it must be possible to store all necessary information about a user-problem.</td>
</tr>
<tr>
<td>S3P7</td>
<td>Personal PMS lacks prompts.</td>
<td>S3GP7</td>
<td>PMS pro-actively solicits problem data.</td>
<td>Any new system should prompt COs for various categories of user-problem details.</td>
</tr>
<tr>
<td>S3P8</td>
<td>Unformatted information in personal PMS.</td>
<td>S3GP8</td>
<td>Formatted problem data.</td>
<td>Any new system will format the structure of problem information.</td>
</tr>
</tbody>
</table>
Table S.4: Transforming users’ problem descriptions to corresponding goal descriptions

<table>
<thead>
<tr>
<th>Problem identifier</th>
<th>Short problem description</th>
<th>Goal identifier</th>
<th>Short goal description</th>
<th>Goal description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1P1</td>
<td>Forgotten problems.</td>
<td>U1GP1</td>
<td>Minimise forgotten problems.</td>
<td>Minimise forgotten problems.</td>
</tr>
<tr>
<td>U1P2</td>
<td>Ignored user-problems.</td>
<td>U1GP2</td>
<td>Minimise ignored user-problems.</td>
<td>Minimise ignored user-problems.</td>
</tr>
<tr>
<td>U1P3</td>
<td>Managers’ problems: priority too low.</td>
<td>U1GP3</td>
<td>Managers’ problems have higher priority.</td>
<td>Managers’ problems have higher priority.</td>
</tr>
</tbody>
</table>
Appendix T

Stakeholder Three’s Goal Hierarchy Diagram

Stakeholder Three’s problem goals

The figure below shows Stakeholder Three’s problem goals.

Figure T.1: Client’s problem goals
Stakeholder Three’s problem goals reorganised

The figure below shows Stakeholder Three’s problem goals and any dependencies between them.

![Diagrams showing Stakeholder Three's problem goals and dependencies.]

Figure T.2: Client’s problem goals

Stakeholder Three’s problem goals and goals

The figure below shows Stakeholder Three’s problem goals and goals.
Figure T.3: Stakeholder Three’s Goal Hierarchy Diagram: Stakeholder Three’s problem goals and goals.
Stakeholder Three’s Goal Hierarchy Diagram: Stakeholder Three’s problem goals and goals and RE’s inferred goals

The figure below shows Stakeholder Three’s problem goals and goals. Normally it would also show new goals inferred by the RE, but in this case no new goals were inferred by the RE.
Figure T.4: Stakeholder Three’s Goal Hierarchy Diagram: Stakeholder Three’s problem goals and goals, and requirements engineer’s inferred goals
Appendix U

Report-1: The COTS system for managing long-term user-problems and goals for a future system

Introduction

In this report I first describe the way that long-term user-problems are brought to the attention of, and managed by, the COTS team. Next I present the views of the client (Jon Ward), the system experts, the Helpdesk manager, and the users on their perceptions of problems with the current management of such long-term user-problems. Following this I present the views of the same set of people on the goals that they would like to see satisfied in any future system for managing long-term user-problems. After this I present some constraints on any computer-based system which might be used to support a system for managing long-term user-problems. Given that constraints are a form of goal and that problem statements may be easily transformed into goal statements (e.g. for the problem x occurs, the goal might be prevent x from occurring), the whole report may be viewed as an expression of goals to be satisfied in any new system for managing long-term user-problems. Of course it may be the case that some of these goals contradict each other. And it will be part of our task to resolve such conflicts in a way which is acceptable to most people. In the final section of the report I outline the actions that I would like you to undertake before our next meeting.

The way user-problems are currently managed by COTS

How do users currently bring their user-problems to the attention of COTS?

Student users experiencing problems with one of the computer systems supported by COTS—Unix, PC, Mac, etc.—nearly always come in person to the Helpdesk in 3P12 and report their problem to one of the Helpdesk per-
sonnel. On rare occasions they might ‘phone in a problem to the Helpdesk or email a problem to the Helpdesk manager. Sometimes a student will report a problem directly to a system expert, e.g. a Unix expert. However, they are not supposed to do this. Staff users sometimes report their computer-system problems in person to the Helpdesk personnel. More often they will phone, email, or visit in person the COTS member of staff who they think is most likely to help them to resolve their problem.

**How many problems are solved on the spot by the Helpdesk?**

A study that I carried out in December 1996 showed, for example, that on one typical day the Helpdesk processed one hundred and twenty-four user-problems brought by one hundred and twelve students, one member of the administrative staff, one lecturer, and nine people from outside of the faculty. Of these one hundred and twenty-four user-problems, ninety-three were of the type which the Helpdesk would normally expect to resolve on the spot, e.g. a student collecting a printout or buying a disc. And these ninety-three were in fact resolved on the spot. Of the remaining thirty-one contingency user-problems—requests for help, advice, information, etc.—the Helpdesk resolved twenty-one on the spot. Of the ten that were not immediately resolved, six were passed on to the Unix experts, two to the Small Systems experts, and two to the Network expert.

**What happens to problems that the Helpdesk cannot resolve?**

User-problems that cannot be resolved by helpdesk personnel are transferred to system experts in a number of ways which are described below and illustrated in the associated diagrams. (See associated description of the notation used.) One method is illustrated in Figure U.1. When the Helpdesk manager determines that the problem cannot be resolved by the Helpdesk staff, she selects the system expert whom she feels is most likely to be able to resolve the problem and communicates the problem to him/her. When the system expert resolves the problem, he/she communicates the details of the resolution back to the Helpdesk manager, who passes them on in turn to the original problem-owner.

In one variation of this process, the Helpdesk manager accompanies the problem-owner on a visit to the system expert where the problem is again recounted. In such cases the helpdesk manager usually remains present while the expert tries to resolve the problem. In this way the spread of knowledge is facilitated. In another method which is illustrated in Figure U.2, the Helpdesk manager communicates the name and location of the selected system expert to the problem-owner, and invites him or her to contact the system expert on their own.

**How many problems are solved on the spot by the system experts?**

The study previously referred to shows, for example, that on one typical day, one of the two groups of Unix experts processed twenty-nine user-problems brought to them by fifteen lecturers, five students, six members of COTS, and three non-faculty staff. Of these twenty-nine user-problems, twenty were
resolved on the spot, one was passed to the other group of Unix experts, one was passed to the group of PC experts, and seven were placed on a list of long-term Unix user-problems.

The next section describes how the system experts manage such long-term user-problems.

How do the system experts manage long-term user-problems?

Introduction

In general each system expert has their own methods for managing long-term user-problems; and these are described in the next section. However, it is worth examining in detail how a typical expert manages such problems.

Figure U.3 shows how a particular expert, after deciding that a user-problem could not be resolved on the spot, adds the problem to his/her set of long-term user-problems. In this case the user-problem is first assigned a priority by the expert. High priority user-problems are written up on a post-it note and stuck to the expert’s workstation screen. Low priority user-problems are treated similarly, but stuck to the border of the workstation screen.
Figure U.4 shows how this system expert schedules long-term user-problems into the day’s work. At the start of the day the set of such problems is reviewed and the day’s schedule is constructed. Then, one at a time, each user-problem is resolved and the solution is communicated to its owner. This subprocess continues until there are no more long-term user-problems.

These descriptions and diagrams are of course more simple than the reality that they describe. For example, they do not say what happens when the expert is unable to resolve a long-term user-problem. Nor do they indicate how the expert manages interrupts from other problem-owners which may occur while he/she is already engaged in resolving a long-term user-problem.

**System experts’ methods for managing long-term user-problems:**

This section describes the methods that each of the system experts used in December 1996 to manage long-term user-problems associated with the system for which they were responsible.
Figure U.3: The system expert resolves user-problem process

Figure U.4: The resolve long-term user-problem process
COTS1:

1. Post-it notes on workstation:

COTS1 uses post-it notes stuck onto the workstation: user-problems of the highest importance are placed upon the workstation screen; those of lower importance are placed around the screen. When a user-problem has been resolved, the stick-it note is removed and an entry is made in COTS1’s log book (see below).

2. Log-book:

COTS1 maintains a log-book in which he records:

- the solution or resolution of user-problems
- tasks to be tackled that day
- changes made during the day

3. Solaris changes

COTS1 maintains a separate log-book for Solaris changes.

COTS2:

1. Memory:

COTS2 normally holds all long-term user-problems in memory. When COTS2 is very busy COTS2 will make an explicit list of problems from memory, representing each problem by a short, descriptive name or phrase.

COTS3:

1. NT: Purpose-built spread-sheet

COTS3 shares a purpose-built spread-sheet with another Computer Officer for managing problems with NT machines. It is used to record the following pieces of information:

- Problem description (short)
- Problem description (detailed)
- Machine
- Machine owner
- Problem priority
APPENDIX U. REPORT-1: LONG-TERM PROBLEMS AND GOALS FOR A FUTURE SYSTEM

2. Unix: ToDoList

For Unix related user-problems, COTS3 uses a "ToDoList" maintained with Emacs. The list has one sentence per user-problem. These sentences are maintained in chronological order.

3. PC: Note-book

For PC related user-problems COTS3 maintains a notebook. In this are recorded notes on PC problems. The notes are updated whenever new information becomes available.

This notebook is also used to record details of 'phone calls related to NT, Unix, or PC user-problems.

COTS4:

1. Flip-pad:

User-problem descriptions are written on an unformatted flip-pad page. The page is crossed through when the user-problem is resolved.

2. Online jobs database

For longer term problems and tasks where it's important to pick up where one last left off after long delays, COTS4 uses a jobs database, which comprises:

- A high level index of jobs, where each job is described by a short, meaningful phrase, and there are pointers to associated free-form areas.

- A job description comprising:

  o Dated progress on job.
  o Job status:
    * A = active
    * B = ongoing
    * C = complete
  o A description of who or what is holding up progress on job.
  o Job started date
  o Job ended date
APPENDIX U. REPORT-1: LONG-TERM PROBLEMS AND GOALS
FOR A FUTURE SYSTEM

- A system for tracing through problem files using a key word search.

COTS5:
------

1. Rely on memory

   COTS5 remembers descriptions of problems that he thinks that can be fixed in a short time.

2. Unformatted note-pad for PC problems:

   COTS5 records details of some PC problems on an unformatted note-pad.

3. Staff email:

   COTS5 asks members of staff to email their problems to COTS5.

4. Purpose-built spread-sheet for NT problems

COTS6:
------

1. Emails

   COTS6’s email mailbox acts as database of user-problems, including those sent by COTS6 to COTS6.

2. Helpdesk user-problem log sheets:

   COTS6 maintains a pile of Helpdesk user-problem sheets. Each is processed in turn and returned to the helpdesk manager when completed.

3. Meeting minutes:

   Fortnightly COTS meetings contain lists of tasks (including user-problems) that have been assigned to various COTS officers including COTS6.

4. Unix: ToDoList

   COTS6 maintains a ToDoList on Unix for holding details of user-problems.
COTS7:
------

1. Solve/resolve:

COTS7 and COTS7’s staff solve on-the-spot most of the problems that users bring to the Helpdesk.

2. Pass problem to CO:

When COTS7 and COTS7’s staff cannot immediately (re)solve a user-problem then they take the problem (and sometimes the user) to a Computer Officer who is an expert in that area.

Problems with current management of user-problems by COTS

This section describes from three points of view—the client’s, the system experts’, and the users’—problems perceived in the way long-term user-problems are managed by COTS. First, the client’s view of such problems is presented.

Client’s view:

Here, the client’s view of the problems associated with COTS management of user-problems.

1. Lost user-problems:

There should be zero "lost" user-problems. User-problems should not be allow to just fade away and eventually be forgotten just because there are no current solutions.

2. Traceability:

Currently most user-problems are communicated to COTS by word of mouth. They are passed around COTS from one member of the team to another in the same way. Consequently, some time after reporting a problem, it is often difficult for a user to find out the current status of that problem. The chain of COTS staff who have been involved with that problem has to be uncovered in order to find the person currently dealing with the problem and thus ascertain its status. This has to be done by asking questions. If one or more people in the chain are not around then it is not possible to complete the construction of the chain.
3. Rate of occurrence unavailable for each type of user-problem:

It is not possible currently to determine the frequency of occurrence of particular types of problems. And in general it’s not possible to look for any kinds of patterns in the occurrence of problems. This is because problem details are not systematically recorded and saved. If this were to be done then it would be possible to search for and identify problem patterns and then be in a position to take more effective measures in dealing with those problems. For example, for a recurring problem, perhaps a helpsheet could be written and given to each user reporting an instance of that problem. This would save time for COTS staff, i.e. the time spent resolving the problem each time it is reported.

4. Solutions unavailable for infrequently occurring but hard-to-fix user-problems:

It is often difficult to retrieve the details of solutions to complex, difficult, infrequently occurring user-problems each time that they are needed. Searching for the solution usually entails a COTS member of staff trying to remember the date when the problem last occurred and then searching through paper-based files in which the details of the solution might have been recorded.

5. Prioritising user-problems:

Computer Officers are often unsure as to how to prioritise a particular user-problem. They each have certain personal criteria for making such decisions, e.g. give top priority to the user-problem that affects most people. And to a certain extent these sets of criteria overlap. However, sometimes they allow their usual criteria to be overridden by other criteria. For example, a member of staff can raise the priority of his or her own user-problem just by being domineering; in such cases the Computer Officers may deal quickly with the user-problem in order to get rid of the problem-owner.

6. Little or no feedback to problem-owners on problem-status:

Currently, COTS are not good at feeding back the current status of user-problems to the problem-owners.

(I suggested that they should be informed of the status of their problem at each significant event in the user-problem’s life-history.)
7. Low visibility of user-problems makes managing difficult:

It is difficult to manage staff who are attempting to resolve user-problems when there is usually no explicit record of each user-problem and its current status to refer to. For example it is currently difficult for the COTS manager to find out which problems are currently being worked on, which problems are blocked and why, which problems are being overlooked, and so on.

8. Problems may remain unsolved indefinitely

A user-problem with a low-priority may remain unsolved indefinitely.

System experts’ and Helpdesk view

In this section the views of each of the systems experts and the Helpdesk manager are presented on problems associated with the way COTS manage long-term user-problems.

**COTS1:**

1. Lost problems:

   COTS1 thinks that some user-problems may be lost.

2. Reactive system:

   COTS1’s system is too reactive in the sense that COTS1 often has to react in real-time to users bringing their problems.

3. Sensitive system

   COTS1’s system is too sensitive to social pressure applied by individuals e.g. some members of staff.

4. Unresolvable problems:

   COTS1 does not think such problems exist.

**COTS2:**

1. Hard to say no:

   COTS2 finds it difficult to say "no" to users, i.e. to turn them away or make them wait when they have problems.
2. Lack of explicit problem stack:

   COTS2 see the lack of an explicit stack of user-problems as a problem, because:
   
   - it leads to some user-problems being forgotten
   - it leads to a low or zero ability to prioritise and schedule problems.

3. Unresolvable problems:

   COTS2 indicated that such problems are placed on the back-burner. These problems may be brought back to life by opportunistic discoveries or events, e.g., finding the source of a required package on the net.

COTS3:

1. Not integrated:

   The three different systems used by COTS3 are not integrated in any way.

2. Lack of user-problem visibility:

   COTS3 cannot see what other Computer Officers are doing who working on related tasks.

3. Lack of user-problem record:

   COTS3 cannot easily tell whether a given user-problem has ever been tackled previously.

4. NT: slow

   The purpose-built spread-sheet is slow to load up.

5. NT: time-consuming to page through

   The purpose-built spread-sheet has to be paged through one page at a time in order to access a particular problem.

6. ToDoList: Insufficient information storable:

   The ToDoList does not enable enough information about a user-problem to be stored.
7. ToDoList: Lacks prompts

The ToDoList does not enable specific categories of information to be prompted for.

8. Notebook: Unformatted

9. Notebook: Lose pages

Sometimes loose pages are lost from the notebook.

10. Unresolvable problems (major):

In the case of unresolvable major problems COTS3 contacts the appropriate technical support team.

11. Unresolvable problems (minor):

Unresolvable minor problems go to the bottom of the list of problems awaiting resolution (usually after some fix or work-around has been installed).

Sometimes opportunistic discovery of relevant information may trigger re-opening of a particular user-problem.

COTS4:

1. Uncertain prioritisation:

Prioritisation is ad hoc; one can never know if right decision was made.

2. Lost problems:

Some problems "drop off the list".

3. Lack of statistics:

No statistics are kept concerning user-problems.

4. Lack of information on time taken to resolve user-problems:

5. Forget information:

Sometimes COTS4 overlooks or forgets to record important information.

6. Unresolvable problems:
Generally these problems remain on the back-burner. Sometimes they solve themselves by going away. For example member of staff has an unresolvable problem, but then leaves the University.

**COTS5:**

1. Problem tracking and recording is difficult:

   Tracking the status of a user-problem can be difficult, particularly if is necessary to go outside the faculty for help.

2. Tracking the status of a problem is particularly difficult when it has become the responsibility of more than one team within COTS.

3. It is not possible to produce accurate statistics showing what work has been accomplished. So for example, it is difficult to find out how much time has been wasted on non-CSM users.

**COTS6:**

1. Unresolvable problems:

   These are placed on the back-burner where they often tend to fade away. Some are given to other Computer officers.

2. Interrupt driven:

   Users may interrupt COTS6 at any time with new problems.

3. Never started:

   Some user-problems which require long-term tasks for their resolution are never started.

**COTS7:**

1. Rudeness from users.

2. Male chauvinism:

   Some male users do not want to communicate their problems
Users’ view

In this section some users’ views are presented of problems associated with the way COTS manage long-term user-problems.

1. Some problems reported by users are ignored and/or forgotten.
2. Managers’ problems are not always given a high enough priority.

Goals for future management of user-problems by COTS

This section describes from three points of view—the client’s, the systems expert’s and the Helpdesk manager’s, and the users’—goals that they would like to see satisfied in any future system for managing long-term user-problems. First, the client’s view of such goals is presented.

Client’s view of goals associated with user-problem management:

This section contains the client's view of the goals of any future user-problem management system.

For some of the goals stated by the client I have extrapolated the goal to generate one or more higher-level goals (marked SGn) to the satisfaction of which it seems to me that it contributes. These higher-level goals can be viewed as providing a rationale for the lower-level ones to which they are linked.

For the client’s goal related to levels of service provision (goal G9), an outline method of satisfying the goal was also suggested. This is included here in the asterisk box.

---

G1. Maintain a database of common problem-solution descriptions and provide easy and efficient access for users to this information. Such a system could be an online system.

SG1: Reduce the number of visits made by users to the Helpdesk or Computer Officers’ offices.
(Goal G1 supports goal SG1)

SG2: Providing more time for COTS staff to spend on solving user-problems.

(Goal SG1 supports goal SG2)

G2. Ensure that user-problems are managed in a uniform way irrespective of their system of origination - PC, MAC, Unix, etc.

G3. In the computer-based component of the user-problem management system use only one set of rules, protocols, forms, etc. for managing user-problems.

(Goal G3 support goal G2)

SG3: Make it easy for users to learn to use the user-problem management system.

(Goal G2 support goal SG3)

SG4: Make it easy for COTS staff to work on user-problems from a variety of system sources.

(Goal G2 support goal SG4)

G4. The priority of each user-problem recorded in the user-problem management system should be automatically raised each time a pre-specified time period elapses.

SG5: Ensure that eventually within a sensible period of time every problem will be tackled.

(Goal G4 supports goal SG5)

G5. It must be possible for a manager to use the user-problem management system to see at any time who is working on what and the status of all current user-problems.

SG6: Manager can act more effectively in managing user-problems.

(Goal G5 supports goal SG6)
G6. User should be informed of the status of his/her problem at sensible times during its lifecycle.

SG1: Reduce the number of visits made by users to the Helpdesk or Computer Officers’ offices.

(Goal G6 supports goal SG1)

SG8: Maximise users’ satisfaction with service provided by COTS.

(Goal G6 supports goal SG8)

G8. It should be possible for users to log faults with the COTS team from a variety of geographical locations, e.g. from the laboratories, from staff offices, from outside the faculty, and from outside the University.

G9. COTS should provide a mechanism whereby each user-problem obtains the level of expertise it requires for its resolution as quickly as possible.

G10. For the life of a problem it is "owned" by only one COTS team member.

This makes it very easy for a user to find out the status of his/her problem since he/she will always be dealing with the same person from COTS. (However, since in practice user-problems are often passed from Computer Officer to Computer Officer, there would need to be a second concept of ownership for COTS use.)

SG9. Make it as easy as possible for users to track their user-problems.

(Goal G10 supports goal SG9)

G11. The User Support team should be housed in the same room.

G12. User Support team members should each become increasingly more able to handle problems from a variety of computer system sources: PC, Unix, MACS, etc.
G13. User Support team members should become increasingly aware of each others’ areas of expertise and skills.

(Goal G11 supports goals G12 and G13)

G14. Although most users are currently "happy" with the service provided by COTS, the level of happiness should be increased until it approaches all users being "very happy".

G15. It must be possible to add details to any record of a problem throughout its lifecycle.

SG10. It must be possible to record, save, and amend the details of problems reported by users. And it must be possible to do this quickly and easily.

System experts’ and Helpdesk view

This section describes the goals that the system experts and the Helpdesk manager would like to see satisfied in any future system for managing long-term user-problems.

COTS1:

1. Stack of user-problems:

   COTS1 would like to work from an explicit stack of problems which have have either been scheduled for COTS1 or scheduled by COTS1.

2. Minimise user contact:

   COTS1 would like less direct involvement with users.

3. Problems via Helpdesk:

   COTS1 would like to receive all user-problems via the Helpdesk.

4. Cut off when necessary:
COTS1 would like to be able to completely terminate contact with users when necessary.

5. Archive of user-problems

COTS1 would find it useful to have a key-word searchable archive of problems since:

- it would help to refresh COTS1’s memory of solutions
- it could be used for sharing knowledge with other staff and new staff

COTS2:

1. Control of problem reception:

COTS2 would like to control the way that user-problems are received, in order to avoid being bombarded with problems. For example, if all user-problems were to be emailed to COTS2 then they could be tackled one at a time.

2. COTS2 would like an assistant - human or computer-based - to prioritise and schedule user-problems for COTS2. However, COTS2 would also like to be able to reschedule these tasks sometimes so that, e.g. COTS2 could select an interesting user-problem to tackle when he wanted to.

3. COTS2 would like to be able to make users aware of when he is free to deal with their user-problems.

4. COTS2 would like to keep a permanent log for two reasons:

   1. It would be personally satisfying to review what has been achieved.
   2. The log might serve as the base of reusable knowledge from telephone numbers and addresses to problem solutions.

COTS3:

1. Integrated, global system:

Any user-problem management system should be integrated and global in the sense that it would support user-problems from all supported computer systems: MACs, PCs, Unix, etc.
2. Easy access to useful services:

   On a user-problem management system it should be easy to obtain the following services:

   - access information on user-problems
   - do searches on user-problems
   - make ad hoc queries about user-problems.

3. High performance:

   A user-problem management system should be high performing.

4. Automatic reminders:

   A user-problem management system should automatically remind the Computer Officers of flagged tasks.

5. Easy to schedule and re-schedule user-problems

   Each Computer Officer should be able to schedule and re-schedule lists of user-problems.

6. Users email/'phone in their problems

   Users should be encouraged to email and 'phone in problems rather than bringing them in personally.

7. Academic staff to visit Helpdesk

   Academic staff should be encouraged to bring their user-problem to COTS via the Helpdesk

**COTS4:**

1. More information available online:

   More information should be available "on tap", e.g. Netwire Magazine "frequently asked questions" should be available online.

2. More time for strategic work:

   Firmer boundaries should be drawn around responsibilities at work to give more time for strategic work. More user-support work could be given to junior Computing Officers and less to senior Computing Officers.
3. Good performance:

Access to a Problem Management System database must take less than 10 seconds.

COTS5:

1. X-Platform:

A User-Problem Management System should be based on an x-platform, e.g. be based on the Web.

2. Quick:

A computer-based User-Problem Management System needs to be quick to use.

3. Central database/ emailed results:

It should be possible for students and staff to log faults and queries to a central database. Such users should be emailed with the resolution of their queries and fault reports.

4. Statistics:

A computer-based User-Problem Management System should be used to produce statistics on problem-solving, e.g, times taken to solve problems, throughput metrics, number of problems reported in unit time, number of problems resolved in unit time.

5. Book COs’ time.

Users (staff and students) should be able to book times to meet COTS support team members.

6. Students and staff should be made aware of exactly what COs can and cannot do within given time-scales.

COTS6:

1. Reduce number of user-problems brought to COTS6

2. Senior COs should only work on long-term tasks.

3. COs should be allowed to work without interrupts.
4. COs should receive user-problems in a controlled way, e.g. only at certain times of the day, and then only by email, for example.

5. COs should have scheduled times for being in interrupt and interrupt-free mode.

COTS7:

1. Not manage consumables

COTS7 would prefer to manage just user-problems at the Helpdesk and not other things like consumables.

2. Employ a second Helpdesk Computer Officer

3. Technical expert on helpdesk

Have one person with wide-ranging technical ability continually present at the Helpdesk

Users’ view:

This section describes the goals that some of the users would like to see satisfied in any future system for managing long-term user-problems.

1. Employ more computer officers

2. Employ more Helpdesk personnel.

3. Enable computer officers to spend more time on problems.

4. Provide more technical training for Helpdesk personnel.

5. Provide customer handling training for Helpdesk personnel.

6. Do not use freshers on the Helpdesk

7. Give priority to academic staff problems.

8. Increase staff coverage for each supported system, e.g. more staff support for MACs

9. Provide Unix and PC support whenever the laboratories are open.
10. Allow users to email problems to Helpdesk personnel and computer officers, i.e. publish relevant email points.

11. Identify a system expert for each system - someone who is prepared to field quite tough problems.

12. Have set times when computer officers would be dedicated to tackling user problems. Publicize them.

13. Publicize skill profiles for each member of the COTS team.

14. Log problems to prevent verbal requests from being lost.

Constraints on any computer-based user-problem management system

This section describes two sets of constraints associated with any future computer-based sociotechnical system for managing long-term user-problems. The first set was given to me explicitly by the client. The second set was gathered by me: while I observed COTS staff carrying out their day to day activities, I asked questions about possibilities for a future computer-based system.

Client’s view:

This section contains the client’s view of the constraints associated with any computer-based user-problem management system.

* Timescales:

  o The bulk of the requirements phase (notwithstanding expected iterations back to this phase during later phases) should be completed by the end of the current academic year, i.e. by September 30th 1998.

  o A complete system should be up and running by the end of the following academic year, i.e. by September 30th 1999.

* Budget:

  o 5-6K pounds, i.e. one placement student (or equivalent) for one year.

* Staff:

  o The introduction of any new system cannot depend on an increase in the number of COTS team members, nor must
it cause any decrease in the number of COTS team members.

* Development methods:
  
o No preferences.

* Development languages:
  
o Since the COTS team will probably maintain the system, it is felt that the language used should be one for which there is currently expertise within COTS. So the candidates are the following:

  - Perl
  - C
  - HTML

* Multi-platform integrated system architecture:
  
o The system should work in a uniform way on a variety of system platforms - Unix, PC, MAC, etc. - and be integrated in the sense that a global view of all user-problems can be accessed and updated from any platform.

* Performance:
  
o The users of the system must be happy with the time and effort that is required to log problems.

  This implies that it will easy for users to enter information quickly. This in turn implies that the user interface will help to minimise typing. So user-interface features might include, for example, click buttons, and lists of problem areas to choose from a menu.

Constraints derived from observations of Helpdesk and system experts

As part of my investigation I spent some time observing COTS staff doing their normal day to day work. During these observation sessions I asked questions about the feasibility of various features of a computer-based system for managing long-term user-problems.

Helpdesk

In the following paragraphs K and M are Helpdesk assistants. I asked K how feasible it would be for her to type in details of problems if a computer-based problem management system were used on the Helpdesk. K says that
she would find typing in a lot of details to be too time-consuming. And that this would be particularly true when she was busy, for example during the lunch hour. K says, after I suggest it to her, that she wouldn’t mind pressing one or two keys for each interaction with a user. She indicates that there might be a problem at shift change-over. I assume that this is because the Help Assistants cannot wait around after a change-over because e.g. they must go to lectures. K agrees, after I suggest it to her, that typing in a user’s system id, while using a computer-based problem management system might be feasible. The id could be used as a key in order to obtain more complete user details. M suggests that to type in a handful of key strokes would be feasible. Thus she agrees with K’s opinion. M suggests that Helpdesk personnel could make notes by hand either during or after a transaction (or both). These could be entered into the system at a later (quieter) time.

**Stakeholder’s task**

Between now and our next meeting I would like you to perform the following actions:

1. Consider how user-problems are currently managed by COTS. (See section 2)

2. Consider the problems perceived with this current management from each different stakeholder’s point of view. (See section 3)

3. Consider the goals that the different stakeholders have proposed for the future management of user-problems by COTS. (See section 4)

4. Try to think of new ways for COTS to manage long-term user-problems which solve the problems presented in section two and satisfy the goals presented in section four. Here you will need to think of the following aspects:

   - How are user-problems to be brought to the attention of COTS staff?
   - How will user-problems be managed within COTS once COTS staff are aware of them?
   - What will be the requirements (i.e. goals) for any computer-based systems that you incorporate in your ideas?
   - What different types of user will need to access any computer-based systems that you incorporate in your ideas?
   - What services will each different kind of user require from any computer-based systems that you need?
   - What data will the computer-based system(s) need to store?
Appendix V

The stakeholders’ response to Report 1

Introduction

This section contains the responses that stakeholders made to the Report 1, “The COTS system for handling long-term user-problems and goals for a future system”. The responses were made by e-mail prior to Stakeholder-Meeting-2. The responses include feedback on the content of the report, either validating it or pointing out anomalies, and suggestions for new served and serving system components that satisfy the stakeholders’ goals. The page numbers and goal designations in the sections below refer to Report 1, which is presented in appendix U starting on page 405.

System Support stakeholder response

Validating Report 1

There probably needs to be a differentiation between the Help Desk and the Computer Officers, rather than just referring to the “COTS team”. For instance, on page 18, Goal 8 should be that the problem can be relayed to the Help Desk from anywhere.

Also, I am not sure that G10 is at all practical, since it assumes that it is assigned to the correct person at the outset, something that cannot be guaranteed. I’d be happier with all jobs being “owned” by the Help Desk, and a record kept of the CO currently assigned to the problem.

Goal 11 is accomplished.

On page 24, Item 7 is probably only acceptable if it relates to something that affects the teaching systems, or it’s a straight choice between something that affects one member of staff and one student.

Item 9 can only be supported as far as providing information to allow users to fix problems for themselves, where possible.

Satisfying stakeholders’ goals

Jobs should be able to be submitted to the Help Desk by any method, with them being communicated to the CO by e-mail when assigned to them. The CO should then check that the assignment is correct, and re-assign if...
necessary. They should then select from a preset list of priority classes, something like:

0 Emergency

1 Infrastructure (i.e. multiple systems affected)

2 Teaching System (i.e. a single system affected)

3 Single-User: Staff

4 Single-user: Student

5 Strategic (i.e. very long term)
Appendix W

Meeting 2: minutes

Write up of notes taken at the stakeholder meeting held on 26/11/97 in 3P5.

Present:
--------

Stewart Green  (Staff stakeholder and meeting facilitator)
Liz Davies  (Helpdesk stakeholder)
Stephen Mullen  (User Support stakeholder)
Heidi Kimber  (User Support stakeholder)
Rick Alsopp  (Student User stakeholder)
Chris Posada  (Student User stakeholder)

Apologies:
---------

Adam Saunders  (Student User stakeholder)
Jin Sa  (Staff User stakeholder)
Phil Naylor  (System Support Stakeholder and Acting COTS Manager stakeholder)
Jon Ward  (Former Client stakeholder and Former COTS Manager stakeholder)

Purpose of the meeting:
-----------------------

The main purpose of this meeting was for the stakeholders to express the various ways they had thought of for meeting the set of stakeholder goals. These goals are described in the report "The COTS system for managing long-term user-problems and goals for a future system" (Report 1). They had received this two weeks previously.
Meeting results: Question one:
-----------------------------

I first asked the stakeholders how difficult they had found the task of trying to create designs to satisfy the goals stated in the report.

Liz Davies:
----------

Liz Davies (Helpdesk Manager) indicated that she had had little time to undertake this work, and had not found the task an easy one.

Liz had found my report to be clear.

Liz had found the goal models that I had drawn to be incomprehensible, simplistic, and annoying.

Chris Posada:
------------

Chris indicated that he had read my report for the first time that morning. But he said that he had understood the report. He indicated that there were too many diagrams; only a summary diagram was required.

Rick Alsopp:
-----------

Rick had found the report clear and easy to follow.

For Rick the diagrams also seemed to help.

Stephen Mullen:
----------------

Stephen said that the report was helpful.

Stephen also felt that there were too many diagrams.

Heidi Kimber:
------------

Heidi also felt that there were too many diagrams.

I did try to find out which of the four types of diagram - problem-goals, problem-goals and goals, restructured problem-goals and goals, or restructured problem-goals and goals plus my extrapolations - had been the most helpful,
but I was unable to elicit this information.

Meeting results: Question two:
----------------------------------

I next asked stakeholders if they had any ideas for ways to satisfy these goals.

Kinds of problems supported:
----------------------------

Liz Davies indicated that from her talks with Heidi Kimber, Stephen Mullen, and Alan Price (User Support), it was felt that a computer-based system should only be used for tracking long-term problems. Heidi and Stephen confirmed this view. A long-term user-problem is one that cannot be solved quickly, on-the-spot, while the user is waiting. The details of such a problem would be entered into a PMS database, the Computer Officer would assign it a priority, but also send the problem and its priority to the COTS manager for review. The COTS manager could change the priority.

I pointed out (representing the absent COTS manager’s viewpoint) that if ALL user-problems were to be logged in some form, then the information (i.e. precise, comprehensive statistics) could be used by a COTS manager, e.g. when negotiating with Faculty management for more resources.

However, LD, HK, and SM did not think that such negotiations, even when backed up with numbers, had ever succeeded for COTS in the past, and also that there was no immediate direct benefit to them from typing in short-term user-problem details.

For these stakeholders (Helpdesk and User Support), one goal seems to be that actions that they undertake must help them directly in succeeding at their immediate tasks. They do not want to do anything that does not have an immediate and direct benefit.

User modes of communicating problems to COTS:
----------------------------------------------

HK and SM indicated that they preferred that all users reported their user-problems directly in person to the Helpdesk. They both felt that a system which involved a user in completing an online form and transmitting it to either the Helpdesk (for subsequent transmission to a User Support person) or to a user...
Support person directly would be inefficient. In most cases they thought that they would have to send a follow-up message to the problem-owner requesting more information. In general they thought that learning about a user’s problem could be achieved more efficiently through face-to-face contact than otherwise. The two students present raised no objection to this, although one did indicate that in certain circumstance it might be useful to be able to e-mail COTS directly; for example when chasing progress made by COTS on a problem that had already been reported.

Problem-solution database:
-------------------------

Both the User Support people and the students felt that it would be a good idea to maintain a database of problem solution pairings. Then a keyword search facility could be used to help find solutions to specific problems.

WWW FAQ:
--------

Both User Support and the students also felt that short-term but frequently occurring problems and their solutions as well as an FAQ document could be accessed from a COTS WWW page linked to the Faculty’s home page.

Platforms:
--------

All present wanted a PMS to be available on all platforms.
Appendix X

Report-2: COTS’ user-problem management system: alternative designs satisfying the stakeholders’ goals

Introduction

A previous report, “The COTS system for managing long-term user-problems and goals for a future system” (November 10th 1997), presented both the problems that stakeholders experience with the current way that user-problems are managed, and the stakeholders’ goals for any new user-problem management system. This report presents a number of socio-technical mechanisms that are intended to attenuate or eliminate the problems, and to satisfy or partially satisfy the goals. Taken together these socio-technical mechanisms may be viewed as constituting a new user-problem management system. A central feature of this new system is the computer-based support that is provided both for reporting and recording user-problems (and eventually their solutions) in a global, integrated database, and for querying the problem-solution database to help to resolve user-problems. This feature is called the computer-based user-problem management system (CBPMS).

Report structure

The report starts out in section two by discussing the new COTS organisation and its likely effect on stakeholders’ goals for a new user-problem management system. This is followed by section three, which defines both “short-term” and “long-term” user-problems, and predicts their likely distribution within COTS. The next section, four, focuses on the requirement to explicitly record the details of user-problems, and introduces the conflict that exists among the stakeholders over whether or not to log short-term user-problems. Section five analyses possible channels of communication between users and COTS and introduces more conflict: this time over whether or not users should be restricted to face-to-face communication of problem details.
A related section (six) follows this; it discusses the main kinds of communication that should occur between COTS and users during the “life” of a user-problem. Section seven discusses how COTS staff (and possibly users) might use the CBPMS to help to resolve user-problems. In section nine the main services to be provided by the CBPMS are presented. Finally, section ten summarises the (hopefully) non-controversial socio-technical mechanisms, reiterates the conflicts over goals, and asks a number of new design questions. This section is intended to be used to “drive” the next stakeholders’ meeting.

Organisation of COTS

This section discusses the impact of the new COTS organisation on certain stakeholder goals.

At the start of my investigation (autumn, 1996) COTS was organised as shown in figure X.1, below:

However, in the summer of 1997, COTS was re-organised. The new structure is shown in figure X.2:

In the new organisation, student users are encouraged to bring their problems to the Helpdesk in the first instance. If the Helpdesk staff cannot resolve a problem, the problem and its owner are then passed on to an appropriate person in User Support selected by Helpdesk staff. If a problem cannot be resolved here, the problem and, optionally, its owner are then passed on to Systems Support. Staff users may communicate their problems either to the Helpdesk, or to User Support, or to Systems Support. Communication of most problems occurs face-to-face, but can be by email or telephone.

The main effect of the new structure is to reduce the number of student user-problems that are dealt with by senior computer officers; and this is also true to a lesser extent of staff user-problems. Such senior computer officers in Systems Support now tend to receive only hard-to-solve user-problems. The writer considers that this reduction is likely to lead to the satisfaction or partial satisfaction of the following stakeholders goals:

(The page numbers in the following tables refer to the page on which a description of the goal or problem occurs in the report “The COTS system for managing long-term user-problems and goals for a future system”.)

Page 19:

Minimise user contact:

COTS1 would like less direct involvement with users.

Page 19/20:

Problems via Helpdesk:

COTS1 would like to receive all user-problems via the Helpdesk.

Page 22:
Figure X.1: COTS organisation: old
Figure X.2: COTS organisation: new
More time for strategic work:

COTS4 thinks firmer boundaries should be drawn around responsibilities at work to give more time for strategic work. More user-support work could be given to junior Computing Officers and less to senior Computing Officers.

---

Senior COs on long-term tasks

COTS6 thinks that senior COs should only work on long-term tasks.

---

To summarise: the new structure is likely to result in senior computer officers having more time available to focus their attention on difficult user-problems, as well as on the strategic issues associated with running COTS. And, since there are no obvious major undesirable consequences resulting from the restructuring of the COTS organisation for any of the stakeholders, it is proposed that no more effort is spent presently on thinking about alternative organisation structures.

User-problems

This section reviews different kinds of user-problems and proposes classifying them into two groups: short-term and long-term user-problems.

User-problems are taken to include anything that causes a staff or student user to interact with a member of COTS staff. The following categories of user-problems have been recognised:

1. Request for help with any aspect of computing.
2. Request for advice on any aspect of computing.
3. Request for information about any aspect of computing.
4. Request to borrow manuals
5. Request for printouts.
6. Request to buy materials (discs, CDs, documents, etc.).

Short-term user-problems

In practice, staff on the Helpdesk, in User Support, and in Systems Support are able to resolve some user-problems on-the-spot, relatively quickly. We will call such problems short-term user-problems.
Long-term user-problems

Other problems take longer to resolve: hours and days rather than minutes. Such problems may be more difficult to resolve, or they may require resources that are not currently available, or the contribution of an expert who is not currently available. We will call such problems long-term user-problems.

Distribution of user-problems within COTS

In the new COTS organisational structure it would be expected that most long-term user-problems would arrive eventually on the desks of the members of Systems Support. However it seems likely that there will also be occasions when User Support staff will take responsibility for dealing with some long-term user-problems. It may also be the case that Helpdesk staff have to deal with such problems; for example, when a CD is being issued to certain students, but supplies of the CD have not yet arrived.

Logging user-problems

This section discusses the need to make explicit records of reported user-problems. It proposes various answers to the following questions: which problems should be logged, how should they be logged, and what details of a problem should be logged?

If we examine some of the problems associated with current user-problem management as perceived by the various stakeholders (see the table below), and also some of the goals that have been requested by the various stakeholders for any new user-problem management system, then it becomes clear that, at least for some, if not for all, user-problems, making an explicit record of the details of such user-problems would help to attenuate the problems and to satisfy the goals outlined in the table.

---

Page 10:
Client:
Lost user-problems:

There should be zero ‘‘lost’’ user-problems. User-problems should not be allow to just fade away and eventually be forgotten just because there are no current solutions.

---

Page 12:
COTS2
Lack of explicit problem stack:

COTS2 sees the lack of an explicit stack of user-problems as a problem, because:

- it leads to some user-problems being forgotten
- it leads to a low or zero ability to prioritise and
schedule problems.

Page 14:
COTS4
Lack of statistics:

No statistics are kept concerning user-problems.

Page 16:
Users

Some problems reported by users are ignored or forgotten.

Page 16:
Client

Facilitate easy access to common problem descriptions:

Maintain a database of common problem-solution descriptions and provide easy and efficient access for users to this information. Such a system could be an online system.

Page 20:
COTS1

Archive of user-problems

COTS1 would find it useful to have a key-word searchable archive of problems since:

- it would help to refresh COTS1’s memory of solutions
- it could be used for sharing knowledge with other staff and new staff

Page 20:
COTS2

COTS2 would like to keep a permanent log for two reasons:

1. It would be personally satisfying to review what has been achieved.

2. The log might serve as the base of reusable knowledge from telephone numbers and addresses to problem solutions.

Page 22:
COTS 5

Statistics:
APPENDIX X. ALTERNATIVE DESIGNS SATISFYING THE
STAKEHOLDERS’ GOALS

A computer-based User-Problem Management System should be used to produce statistics on problem-solving, e.g., times taken to solve problems, throughput metrics, number of problems reported in unit time, number of problems resolved in unit time.

Page 24:

Users

Log problems to prevent verbal requests from being lost.

---------------------------------------------

It seems clear that if an explicit record of user-problems is made and kept, i.e., user-problems are logged, then it will be easier to attenuate or eliminate the problems outlined in the table above, and to satisfy or partially satisfy the goals outlined in the table.

However, some important questions remain to be answered:

1. Which user-problems will be logged?
2. How will user-problems be logged?
3. What details of user-problems will be logged?

These questions are addressed in the following sections.

Which user-problems will be logged?

All stakeholders agree that it would be sensible and feasible to log long-term user-problems. This is because once logged they would be difficult to lose; they could be prioritised; related statistics could be generated; and they could be used as an information resource by COTS staff and users, which would be particularly useful if it included corresponding solution information.

Thus for any new user-problem management system one possibility would be to provide a means for logging long-term user-problems, and a policy of mandating all COTS staff to log such problems.

However, the situation with short-term user-problems is less clear-cut.

On the one hand the client stakeholder would like to have easy access to comprehensive, quantitative information about user-problem management. For example, it would be as useful for the client to know how many times printouts are collected by users, as to know how many long-term problems are solved in a week. This is because, for example, such information can be used by a COTS manager in meetings with the faculty Dean, to press the manager’s case for the provision to COTS of more resources. This requirement of the client stakeholder is expressed both in problems he has recognised, and in goals he has requested to be satisfied in any new user-problem management system (see below).

---------------------------------------------
Comprehensive user-problem management reports may be produced

All User Support actions must have immediate, direct benefits for User Support

Record short-term user-problem details

Figure X.3: Goal conflict over recording short-term user-problem details

Client

Rate of occurrence unavailable for each type of user-problem:

It is not possible currently to determine the frequency of occurrence of particular types of problems. And in general it’s not possible to look for any kinds of patterns in the occurrence of problems. This is because problem details are not systematically recorded and saved. If this were to be done then it would be possible to search for and identify problem patterns and then be in a position to take more effective measures in dealing with those problems. For example, for a recurring problem, perhaps a helpsheet could be written and given to each user reporting an instance of that problem. This would save time for COTS staff, i.e. the time spent resolving the problem each time it is reported.

On the other hand, at the second meeting of the stakeholders (November 12th 1997), a number of User Support staff indicated that they should not have to record the details of short-term user-problems, because this would be a time-consuming activity which would have no direct benefit to them; for example making such records would not improve their efficiency. In this view they seemed to be expressing implicitly the following goal: any action undertaken by User Support should help them directly to succeed at their immediate tasks.

This conflict between the view of the client stakeholder (ex-COTS manager) and the view of the User Support stakeholders is represented in figure X.3:

In response to the recognition of the conflicting views of some stakeholders on this issue an attempt will be made to achieve a consensus or accommodation. To facilitate this attempt we will consider here four ways for managing short-term user-problems, as follows:
1. All short-term user-problems are to be logged in exactly the same manner as long-term user-problems.

2. Some, but not all, short-term user-problems are to be logged in the same way as long-term user-problems. The remainder are not to be logged.

3. A minimal amount of information on short-term user-problems is to be logged, e.g. user-problem type only.

4. No information is to be logged on short-term user problems.

Clearly, option three may be combined with either option one or option two.

Presumably, adopting option four would completely satisfy User Support staff, but not the client. On the other hand, adopting option one would presumably completely satisfy the client, but not User Support staff. Perhaps a compromise might be acceptable to both groups of stakeholders: for example, adopting options one and three, or two and three. An attempt to resolve this issue will be made at the next meeting of the stakeholders.

It is worth noting here that the client also expressed the following performance constraint:

Page 25:

Client

Performance:

The users of the system must be happy with the time and effort that is required to log problems.

This implies that it will easy for users to enter information quickly. This in turn implies that the user interface will help to minimise typing. So user-interface features might include, for example, click buttons, and lists of problem areas to choose from.

This shows that the client is concerned that any new user-problem management system should be practicable for all its direct hands-on users to use.

**What details of user-problems will be logged?**

**Long-term user problems**

For long-term user-problems the information to be logged might include the following:

- Problem owner’s name, email address, and telephone number (optional).
• Date and time the problem is logged. (Automatically supplied in any computer-based system.)

• Problem description: this data field might be organised in either a free-form manner, a pre-formatted manner, or a mixture of both. The last option is illustrated below:
  – Problem type: (information request, advice request, problem report, manual borrowing, etc.)
  – Problem location:
  – Machine:
  – Software package:
  – Problem description: free form

During the course of a user-problem’s “life” within COTS, other information would need to be logged with each problem, this might include the following:

• Problem priority (see section eight)

• Person assigned to problem:

• Date problem assigned:

• Problem status: (blocked, closed, in queue, etc.)

• Date problem closed:

• Solution details:

A discussion of the “solution details” information is included in a later section (see “Resolving User Problems”).

**Short-term user problems**

At the next meeting of the stakeholders an attempt will be made to reach a consensus over which details of short-term user-problems will be logged. As has been pointed out above, one possibility would be to log just the type of user-problem as follows:

• Problem type: (information request, advice request, problem report, etc.)

**How will user-problems will be logged?**

Let us review some of the problems expressed by the various stakeholders and likewise some of the goals requested by various stakeholders, both relating to the question of how user problems will be logged. These are listed in the table below:
Facilitate easy access to common problem-solution descriptions

Maintain a database of common problem-solution descriptions and provide easy and efficient access for users to this information. Such a system could be an online system.

Page 20:
COTS 1

Archive of user-problems

COTS 1 would find it useful to have a key-word searchable archive of problems since:

- it would help to refresh COTS 1's memory of solutions
- it could be used for sharing knowledge with other staff and new staff

Page 20:
COTS 2

COTS 2 would like to keep a permanent log for two reasons:

1. It would be personally satisfying to review what has been achieved.

2. The log might serve as the base of reusable knowledge from telephone numbers and addresses to problem solutions.

Page 23:
COTS 3

Easy access to useful services:

On a user-problem management system it should be easy to obtain the following services:

- access information on user-problems
- do searches on user-problems
- make ad hoc queries about user-problems.

Page 22:
COTS 5

Central database/ emailed results:

It should be possible for students and staff to log faults and queries to a central database. Such users should be emailed with the resolution of their queries and fault reports.
APPENDIX X. ALTERNATIVE DESIGNS SATISFYING THE STAKEHOLDERS’ GOALS

Maximise efficiency of comprehension of user-problems by COTS staff

Improve control over reception of user-problems

Minimise effort required by users to communicate user-problems

Maximise face-to-face communication of user-problems

Encourage multi-modes of communication: face-to-face, e-mail, phone, CBPMS

Figure X.4: Goal conflict over allowed channels of communication

Although, in theory, user-problems could be logged using a distributed paper and pencil system, or a distributed online flat-file system, or a set of independent database systems, some of the goals above imply that a single, integrated, global computer-supported database should be used to log all user-problems; for example, the last one in the list has that implication. It should be possible to search such a database and to submit to it ad hoc queries.

How will users communicate their problems to COTS?

The earlier section on the internal organisation of COTS described the ways in which users are currently encouraged to communicate their problems to COTS staff. However, as can be seen by examining in the table below the goals relevant to this issue expressed by the various stakeholders, some stakeholders would like to maximise face-to-face communication of problems to the Helpdesk staff in the first instance, while others would like to allow other forms of communication (email or telephone) to a wider variety of COTS staff.

This conflict of views between the current User Support and Helpdesk stakeholders on the one hand, and the COTS manager, some senior Computer Officers, and some users on the other hand is represented in figure X.4:

It is worth noting there that the same stakeholder may on occasion express apparently conflicting goals. For example, the first two goals in the table below expressed by COTS3 seem to conflict.

<table>
<thead>
<tr>
<th>G1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users email/’phone in their problems</td>
</tr>
</tbody>
</table>
Users should be encouraged to email and 'phone in problems rather than bringing them in personally.

G2
Page 21:
COTS3

Academic staff to visit Helpdesk

Academic staff should be encouraged to bring their user-problem to COTS via the Helpdesk

G3
Page 22:
COTS5

Central database/ emailed results:

It should be possible for students and staff to log faults and queries to a central database. Such users should be emailed with the resolution of their queries and fault reports.

G4
Page 23:
COTS6

COs should receive user-problems in a controlled way, e.g. only at certain times of the day, and then only by email, for example.

G5
Page 24:
Users

Allow users to email problems to Helpdesk personnel and computer officers, i.e. publish relevant email points.

G6
Page 18:
Client:

It should be possible for users to log faults with the COTS team from a variety of geographical locations, e.g. from the laboratories, from staff offices, from outside the faculty, and from outside the University.

G7
User support (meeting 26/11/97)
All users - staff and students - should report their problems in person directly to the Helpdesk in the first instance. This is because problems are communicated most efficiently by face-to-face verbal interaction, i.e. discourse. Online form-filling and emails are likely to be inefficient due, for example, to the probable need for several emails to be sent to elicit all the information about a problem to proceed with tackling it.

G8
Student users (a view expressed at the second stakeholders’ meeting 26/11/97)

It would be useful to be able to email COTS officers directly, e.g. when chasing progress made by COTS on previously reported problems.

G9
Student users (a view expressed in the user questionnaire returns)

It would be useful for students if COTS were to publish a list of staff members, each member’s technical expertise, and an email address to which problems could be directly mailed.

G10
Page 19 - 20
COTS1

Problems via Helpdesk:

COTS1 would like to receive all user-problems via the Helpdesk.

G11
Page 20:
COTS2

Control of problem reception:

COTS2 would like to control the way that user-problems are received, in order to avoid being bombarded with problems. For example, if all user-problems were to be emailed to COTS2 then they could be tackled one at a time.

Clearly, in order to create a new user-problem management system that will be widely used by COTS and users, it will be necessary to eliminate or at least reduce this conflict. An attempt will be made to do this at the next stakeholders’ meeting. It seems likely that it will prove easier to reach a consensus or accommodation at that meeting if the alternatives are clearly
laid out. In the paragraphs below I have attempted such an analysis.

First, any new user-problem management system may need to introduce and encourage the use of two distinct communication protocols: one for staff users and one for student users.

Second, the target for communicating user-problems may be one of the following in any new system: Helpdesk staff, User Support staff, and Systems Support staff.

And third, the channel of communication might be one of the following:

1. face-to-face
2. telephone
3. email
4. computer-based user-problem management system (CBPMS)

The intended operation of the first three channels listed above should be self-evident. However, the intended operation of a possible CBPMS needs further explanation.

It is intended that users would be able to use a CBPMS to call up an online form which would prompt them to supply details of their problems. After completing a form, a user would be able to send it online to COTS.

**Communication alternative one**

Presumably, in any new user-problem management system, User Support and Helpdesk stakeholders would like both staff and student users to communicate their problems in person to helpdesk staff in the first instance.

**Communication alternative two**

Other stakeholders might like to restrict student users to communicating their problems to the Helpdesk but might not be concerned whether this was done face-to-face, over the ’phone, by email, or by a CBPMS. This group of stakeholders would be prepared to allow staff users to choose to whom they report their problems - Helpdesk, User Support, or Systems Support - and how to report them - face-to-face, ’phone, email, or CBPMS.

**Communication alternative three**

Yet other stakeholders might like to restrict student users as follows: face-to-face or ’phone communication of problems must be to the Helpdesk, but problems may be communicated by email or CBPMS directly to the Helpdesk, User Support, or Systems Support. Again, staff would be allowed to communicate problems to any member of COTS staff in any way they wanted.

There may be other communication permutations which one or more stakeholders would favour. At the next meeting of the stakeholders it will be important to try to derive a consensus on the modes of communication to be allowed in any new user-problem management system.
Publishing details of COTS staff

One implication of allowing and encouraging users to target individual COTS members of staff, either in person, by 'phone, by email, or by CBPMS, would seem to be that various details about each COTS member of staff should be published widely, or be made easily accessible, or both. The details might include the following:

- Name of COTS staff member:
- COTS group: (Systems Support, User Support, or Helpdesk)
- Area of expertise: (Unix, PC, Mac, Oracle, Network, Book borrowing, Printouts, Sales, etc.)
- Email address
- Room number
- Telephone number:
- Preferred mode of contact: (Face-to-face, email, telephone, or CBPMS)
- For face-to-face, preferred times of contact:

The extent to which the initial-communication alternatives satisfy the stakeholders’ goals

It is useful to consider the extent to which each of the three communication alternatives described above supports or undermines stakeholders’ goals for initial communication of user-problems to COTS:

- Communication alternative one (CA1).
  - Supports: G2, G7, G10.
  - Undermines: G1, G4, G5, G6, G8, G11.

Clearly, CA1 supports goal G7 proposed by User Support stakeholders, G10, proposed by a Systems Support stakeholder, and G2 proposed by COTS3. On the other hand, CA1 provides no support for the achievement of goals G1, G4, and G11 suggested by COTS staff stakeholder, goal G6 proposed by the client stakeholder, and goals G5 and G8 proposed by user stakeholders.

- Communication alternative two
  - Supports: G1, G4, G5, G6, G8, G11.
  - Undermines: G2, G7, G10.

CA2 has exactly the opposite effect to CA1 on satisfying stakeholders’ goals.

- Communication alternative three
  - Supports: G1, G4, G5, G6, G8, G11.
  - Undermines: G2, G7, G10.

CA3 has the same effect as CA2 on goals explicitly stated by the stakeholders.
Other effects of the initial-communication alternatives

Possible solution components, like the three initial-communication components described above, may be evaluated with respect to a set of goals explicitly expressed by the stakeholders. It is also possible for the stakeholders to review each solution component and to try to predict outcomes, other than those related to the explicitly stated goals, which might result from the adoption of the solution component. Each outcome might then be evaluated for its likelihood of occurrence by each of the stakeholders, and also assessed as a positive or a negative effect from each individual stakeholder’s point of view.

As an example of this approach the writer has tried to predict additional effects which might be associated with “communication alternative one” and “communication alternative two”.

Writer’s prediction of additional effects for communication alternative one

1. Some staff may violate the rule and visit selected computer officers directly rather than going through the Helpdesk. This may lead to anger and frustration in COTS staff.

2. There may be a high number of interactions at the Helpdesk relative to the number associated with communication alternative two. Thus there may be a high helpdesk work-load compared to the other alternative.

3. There may be a high number of Helpdesk to User Support interactions, and User Support to Systems Support interactions relative to communication alternative two before a user-problem is matched with the an appropriate level of expertise.

4. There may be fewer interactions with a user needed before a user-problem is understood well enough to start work on it relative to communication alternative two.

5. Relative to communication alternative two, there may be a higher number of real-time interrupts for User Support and Systems Support staff caused by other COTS staff bringing problems and their owners to individuals in these groups.

Writer’s prediction of additional effects for communication alternative two

1. There may be a low number of interactions at the Helpdesk relative to the number associated with communication alternative one. Thus there may be a low helpdesk work-load compared to the other alternative.

2. There may be a low number of Helpdesk to User Support interactions, and User Support to Systems Support interactions relative to communication alternative one before a user-problem is matched with the an appropriate level of expertise.
3. There may be more interactions (via email or a CBPMS) with a user needed before a user-problem is understood well enough to start work on it relative to communication alternative one.

4. Relative to communication alternative one, there may be a lower number of real-time interrupts for User Support and Systems Support staff caused by other COTS staff bringing problems and their owners to individuals in these groups.

5. Users may be happier with the service provided as the increase in the number of available different communication channels makes it easier for them to communicate their problems.

How will COTS staff communicate with users?

This section describes the main communications that should occur between COTS and users throughout the “life” of a problem. The section assumes that users will be allowed to communicate their problems to COTS staff face-to-face, by telephone, by email or by CBPMS. However in any final system not all these options may be permitted as has been discussed above.

The only stakeholder goal concerning this subject was expressed by the client stakeholder and is tabulated below:

<table>
<thead>
<tr>
<th></th>
<th>Page 11:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Little or no feedback to problem-owners on problem-status:</td>
</tr>
<tr>
<td></td>
<td>Currently, COTS are not good at feeding back the current status of user-problems to the problem-owners.</td>
</tr>
</tbody>
</table>

The three main sets of circumstances in which communication with a user by a member of COTS is appropriate are described below:

1. When a user initially communicates a problem to COTS.

2. When a significant event occurs in the “life” of a long-term problem.

3. When a problem is resolved by COTS.

Each of these circumstances is now considered in turn.

Initial communication

Face-to-face

If a problem is communicated by a user in person to either the Helpdesk, User Support, or to Systems Support, then a verbal acknowledgement of receipt of the problem will be supplied by the COTS member of staff to the user. The COTS member of staff will then try to resolve the problem.
If the problem is resolved on the spot, the solution details will be communicated verbally to the user straight away. If it cannot be resolved then normally it will be passed to some one else within COTS. In this case the name and location of the person to whom the problem has been transferred will be given verbally to the user.

If the problem is not solved on the spot nor passed to some one else, then this will normally mean that the problem has been recognised as a long-term problem which will be dealt with, i.e. scheduled and tackled, by the COTS staff member currently handling the problem. The user will be told that the problem is a long-term one; the user will also be given the name of the COTS member of staff dealing with it; the user will also be told that they will be informed by email or by ’phone (for staff users) of significant changes in the status of the problem as and when such changes occur.

**Telephone**

The procedures just described for face-to-face encounters will be followed for problems communicated to COTS staff over the ’phone.

**Email**

If a problem is communicated to the Helpdesk staff, or to User Support staff, or to Systems Support staff by email then the COTS member of staff receiving the email will read the email and take one of the following actions:

1. If they can resolve the user’s problem they will email the user with the solution details.

2. If they cannot resolve the user’s problem but recognise that another member of COTS staff would probably be able to resolve it, then they will forward the email to the selected COTS staff member. They will also reply to the user acknowledging receipt of the problem and indicating who, within COTS, is currently dealing with it.

3. If they recognise that they are the appropriate person to deal with the problem, but that it is a long-term problem, then they will reply to the user’s email informing them that the problem has been received, supplying their own name, and informing them that they will be notified by email or by ’phone whenever the status of the problem changes.

**Communication of long-term user-problems face-to-face or by email or by ’phone**

In the face-to-face, ’phone, and email contexts, as soon as a member of COTS accepts responsibility for managing the problem as a long-term problem, they will enter its details on the CBPMS database. Entering the details will include entering a priority for the problem. (See also the later section “Managing Problem Priorities”).
Communication via a computer-based user-problem management system (CBPMS)

A problem may be communicated to either Helpdesk staff, or User Support staff, or Systems Support staff via the CBPMS. In this case the CBPMS will add the problem details to the CBPMS database and automatically email the targeted member of COTS staff, notifying them of the existence of the problem.

The targeted member of COTS staff will read the problem details and take one of the following actions:

1. If they can resolve the user’s problem they will email the user with the solution details. They will also update the problem record on the CBPMS database with the solution details.

2. If they recognise that another member of COTS would be able to resolve the user’s problem, then they will use the CBPMS to add the selected person’s name to the problem record as the person currently dealing with the problem. This will automatically cause an email to be sent to the selected person notifying them that a new problem has been assigned to them. The former COTS staff member will also email the user acknowledging receipt of the problem and indicating who within COTS is dealing with it.

3. If they recognise that they are the appropriate person to deal with the problem, but that it is a long-term problem, then they will email the user informing them that the problem has been received, supplying their own name, and informing them that they will be notified by email or by ’phone whenever the status of the problem changes.

The patterns of communication described above are summarised in figure X.5 below:

1. User communicates problem to someone in COTS, e.g. COTS-a.
Either 2 \( \text{COTS-a communicates to the user the solution to this short-term problem.} \)

Or 3 a \( \text{COTS-a transfers problem details to some one else in COTS e.g. COTS-b.} \)

\( \text{and} \)

3 b \( \text{COTS-a communicates to the user the name of the member of staff currently dealing with their problem, i.e. COTS-b.} \)

Or 4 a \( \text{COTS-a retains responsibility for resolving this long-term user-problem.} \)

\( \text{and} \)

4 b \( \text{COTS-a informs the user of his or her own name, i.e COTS-a.} \)

**Significant event occurs**

For long-term problems that have been logged on the CBPMS and are currently the responsibility of one COTS officer, that COTS officer will inform the user by email or by ’phone whenever there is a change in the status of the problem.

Examples of such status messages might include the following:

- Your problem has been received, prioritised, and is queued awaiting attention.
- Your problem is receiving attention now.
- Work on your problem is blocked awaiting the following resource: (software name, hardware name, person name, money, other)
- Work on your problem has restarted.

**User-problem resolved**

When a long-term problem has been resolved then the user should be informed immediately by email or by ’phone of the resolution details.

**Summary**

It is expected that if the communication protocols outlined above are adhered to then the client’s goal relating to feeding back problem status information to the users (see above) will be satisfied. In addition it is expected that other benefits, particularly for the users, will result from these protocols. For example a user will always know who within COTS is dealing with their problem at all times.

**Using the CBPMS to help to resolve user-problems**

It is clear from the problems and goals expressed by COTS stakeholders (ex-manager and staff), given in the table below, that finding and re-using the solution details for recurring problems is often a time-consuming procedure, and one which may frequently yield no results.
Solutions unavailable for infrequently occurring but hard-to-fix user-problems:

It is often difficult to retrieve the details of solutions to complex, difficult, infrequently occurring user-problems each time that they are needed. Searching for the solution usually entails a COTS member of staff trying to remember the date when the problem last occurred and then searching through paper-based files in which the details of the solution might have been recorded.

Lack of user-problem record:

COTS3 cannot easily tell whether a given user-problem has ever been tackled previously.

Client

Maintain a database of common problem-solution descriptions and provide easy and efficient access for users to this information. Such a system could be an online system.

COTS1 would find it useful to have a key-word searchable archive of problems since:

- it would help to refresh COTS1's memory of solutions
- it could be used for sharing knowledge with other staff and new staff

COTS2 would like to keep a permanent log for two reasons:

1. It would be personally satisfying to review what has been achieved.
2. The log might serve as the base of reusable knowledge
from telephone numbers and addresses to problem solutions.

One way of attenuating or eliminating these problems and satisfying or partially satisfying these goals would be as follows: first, maintain details of solutions alongside problem descriptions in the CBPMS database; and, second, allow COTS staff, and perhaps users, to query this problem-solution information. Such queries might include pre-formatted inquiries as well as ad hoc inquiries.

It is possible to imagine that in the process of resolving user-problems, particularly long-term ones, information relevant to a solution slowly accumulates in a relatively haphazard way. Such information might include, for example, telephone numbers, machine names, file names, tasks that have been undertaken, and so on.

In order for such solution information to be more effectively re-used, its haphazard form would need to be transformed into a well structured, coherent form. Thus it is suggested that after a member of COTS staff has resolved a long-term user-problem, written to the problem-owner, and closed the problem, they should rewrite the problem solution on the CBPMS database so that it is well-structured and coherent.

The assignment of priorities to, and the scheduling of, user-problems

Once a user-problem has been identified by a member of COTS staff as long-term, and has been accepted by that member of staff as their responsibility, then they will assign a priority to the problem by adding the assigned value to the problem record on the CBPMS database.

Priority levels and their codes will be as follows:

1. Emergency
2. Staff manager affected
3. Multiple systems affected
4. Single system affected
5. Single staff user affected
6. Single student user affected
7. Very long-term strategic problem

COTS members of staff will assign priorities to long-term problems according to the criteria listed above.

The COTS manager may alter the priority assigned to any user-problem logged in the CBPMS database. When such action is taken, the CBPMS will automatically send an email to the officer dealing with the problem informing them of the change in priority.
When a user-problem has been held in the CBPMS database for a certain period of time without being tackled then its priority will be increased by the CBPMS automatically. The priority will continue to be incremented periodically in this way. I suggest that after each week in the system one is deducted from a problem’s priority code.

When a COTS member of staff requests the CBPMS to schedule user-problems that have been assigned to him or her, the CBPMS will present the list of problems in priority order.

The writer predicts that if the CBPMS provides the services described above and if the protocols just outlined are followed, then the following goals of the stakeholders will be satisfied, either in full or in part.
APPENDIX X. ALTERNATIVE DESIGNS SATISFYING THE
STAKEHOLDERS’ GOALS

COTS4

Uncertain prioritisation:

Prioritisation is ad hoc; one can never know if right decision was made.

Page 15:

COTS6

Never started:

Some user-problems which require long-term tasks for their resolution are never started.

Page 16:

Users

Some problems reported by users are ignored and/or forgotten.

Page 16:

Users

Managers’ problems are not always given a high enough priority.

Page 17:

Client

The priority of each user-problem recorded in the user-problem management system should be automatically raised each time a pre-specified time period elapses.

Page 19:

COTS1

Stack of user-problems:

COTS1 would like to work from an explicit stack of problems which have have either been scheduled for COTS1 or scheduled by COTS1.

Page 20:

COTS2

COTS2 would like an assistant - human or computer-based - to prioritise and schedule user-problems for COTS2. However, COTS2 would also like to be able to reschedule these tasks sometimes so that, e.g. COTS2 could select an interesting user-problem to tackle when he wanted to.
COTS3

Easy to schedule and re-schedule user-problems

Each Computer Officer should be able to schedule and re-schedule lists of user-problems.

Staff users

Give priority to academic staff problems.

Using the computer-based problem management system

In this report a CBPMS has been proposed. It is useful to summarise in outline what services will be provided by the CBPMS for each distinct group of CBPMS users. In the next report these outline descriptions will be elaborated into a set of full-blown “use cases” from which it is expected that design and implementation will proceed.

CBPMS users

Currently, it seems necessary to define three distinct groups of direct users of the CBPMS as follows:

1. Users
2. COTS staff
3. COTS manager
4. CBPMS administrator

CBPMS services

The services to be provided for each of the three groups are now outlined in turn:

- Users
  1. Start CBPMS session
  2. Send problem details to COTS
  3. Query details of COTS experts
  4. Query problem-solution database (possibly)
  5. Print from problem-solution database (possibly)
  6. End CBPMS session
APPENDIX X. ALTERNATIVE DESIGNS SATISFYING THE
STAKEHOLDERS’ GOALS

- COTS staff
  1. Start CBPMS session
  2. Prioritise a problem
  3. Accept responsibility for problem
  4. Pass problem on to another member of COTS staff
  5. Update problem details
  6. Add/modify solution description
  7. Query problem-solution database
  8. Print from problem-solution database
  9. End CBPMS session

- COTS manager
  1. All services available to COTS staff plus:
  2. View online pre-formatted management reports
  3. Print pre-formatted management reports
  4. View online ad hoc management reports
  5. Print ad hoc management reports

- CBPMS administrator
  1. Start up CBPMS
  2. Shut down CBPMS
  3. All other operations

Issues for the next stakeholders’ meeting

This section summarises key components of a new user-problem management system. It highlights both components over which there is consensus among the stakeholders, and components over which there is conflict. In addition the section poses some new questions.

At the next stakeholders’ meeting I intend to ask them whether the non-controversial components have been correctly described, and whether any areas have been overlooked. In addition I will try to help the stakeholders to reach a consensus in the areas of conflicting requirements.

Validating the non-controversial socio-technical mechanisms

The key components of a new user-problem management system are as follows:

- COTS will be organised into System Support, User Support, and a Helpdesk.

- COTS staff will use a new user-problem management system to manage user-problems.

- A central feature of the new system will be a global, integrated, computer-supported user-problem database.
• Descriptions of all long-term user-problems will be recorded in the database.

• Descriptions of the solutions associated with long-term user-problems will also be recorded in the database.

• The priority of each user-problem will be raised automatically after a predetermined period of time.

• COTS staff will use the problem-solution information in the database to help to resolve user-problems.

Resolving conflicts over goals
The main areas of conflict among stakeholders are as follows.

1. Should details of short-term user-problems be logged in the database? And if so, what details exactly?

2. How should users report their user-problems to COTS? Always face-to-face? Or multi-mode: face-to-face or by telephone or by email or by computer-based problem management system? Should staff and student users be governed by different rules in this area? And if so, what rules?

Outstanding questions
The main questions still to be answered by the stakeholders are as follows:

• Should the users have access to the problem-solution information held in the CBPMS database?
Appendix Y

Stakeholder-Meeting-3

Stakeholders’ feedback on Report 2

The following feedback was received from the System Support Stakeholder on Report-2 prior to Stakeholder-Meeting-3.

General:

Page 28:

In section 6.4, I feel that the communication protocols should be implemented within the system using e-mail, rather than relying on the cooperation of COTS.

Page 30:

Item 2, what is a Staff manager?

Resolving conflicts:

1.

Short term problems should be recorded, at the very minimum as a common problem type, to allow for the collection of statistics. This would also allow problems to be bundled together, with a common solution (or things to try).

The ACOs and Helpdesk staff need to realise that they aren’t being expected to spend 10 minutes recording information on everyone who (for instance) can’t log in, but spending 15-20 minutes on the first case will mean that all the rest can be dealt with much more quickly, or the users can even sort themselves out. It should be borne in mind that none of the Helpdesk assistants have worked with us for more than a year, and neither have Stephen or Heidi. They do not realise that the same things will keep coming up each year, ad infinitum, and that it does not make sense to deal with each case from scratch. As a possible compromise, the short term problems could be logged against a common type, and
a report prompted for once a certain number of problems of that type have been entered. This would save spending 20 minutes recording a problem which turns out to be a one-off.

2.

I prefer communications model 2. Students need to report by some mechanism whereby they can be forced to supply all the information needed to start work on problem. Contacting a Computer Officer and saying "The Web doesn’t work" is not going to do anyone any good. This could mean making them report via the Helpdesk, or the PMS needs to have "required" fields without which a problem cannot be submitted.

Whatever mechanism we put in place, staff will usually go straight to whichever they think is the correct Computer Officer.

Outstanding questions:

I feel the problem-solution information should be made available to users - there are plenty of problems they could solve for themselves, if the information was presented to them in an accessible way. There are still those who would insist on having their hands held though.
Agenda

Meeting of the stakeholders for the user-problem management system:

Date: 18/03/98

AGENDA

1. Introduction

2. Resolving conflict
   2.1 Recording short-term user problems
   2.2 Modes of communication between users and COTS

3. Validating the report
   3.1 COTS organisation structure
   3.2 User-problems
   3.3 Logging user-problems
   3.4 Users communicating with COTS
   3.5 COTS communicating with users
   3.6 Solving user-problems with the CBPMS
   3.7 Prioritizing user-problems

4. Futures
   4.1 Short-term
   4.2 medium-term
   4.3 Long-term

5. AOB
Conflict papers

Short-term user-problem conflict

Conflict 2.1 recording short-term user-problems:

------------------------------------------------

Issue:
------

Some stakeholders, e.g. the COTS manager, would like the user-problem management system to record short-term user-problems. They think that this data is required to manage user-problems effectively.

On the other hand, some stakeholders, e.g. Helpdesk staff, think that recording such data would take too long and be without any immediate benefit to their performance of current tasks.

Resolution possibilities:
------------------------

1. Record no details of short-term user-problems on the CBPMS.

2. Record the same pre-specified information about each pre-specified user-problem type (or COTS-user interaction).

3. Record pre-specified but possibly varying information about each user-problem type.

Help request: Problem type + email id. + problem description + solution
Information: Problem type + problem description + solution
Advice: Problem type + problem description + solution
Printouts: Problem type
Manuals: Problem type
Buy: Problem type

4. Phil Naylor has suggested a scheme like 3 above. However in his scheme only the problem type would be recorded a particular "Help request" until a certain number, say 5, similar requests had occurred and then the problem description and solution would be recorded too.
User-COTS communication conflict

Conflict 2.2 Users communicating with COTS:
-------------------------------------------

Issue:
------

Some stakeholders feel that face-to-face reporting of user-problems directly to Helpdesk staff in the first instance is the most efficient way for a user to communicate their problem to COTS.

Other stakeholders would like to permit users to communicate their user-problems to COTS staff in other ways, e.g. by email.

It is probably the case that academic staff users will use whatever means are at their disposal to communicate their user-problems to whoever they think is most likely to fix them.

Resolution possibilities:
------------------------

The report presents three alternative communication protocols: CA1, CA2, and CA3. CA2 and CA3 both reflect the reality of patterns of interaction likely to be exhibited by academic staff. CA2 allows student users to choose from a number of ways of reporting a user-problem: face-to-face, telephone, email, or CBPMS. However, in each case, the target is always the Helpdesk in the first instance.
Validation questions

Meeting of the stakeholders for the user-problem management system:
---------------------------------------------------------------

Date: 18/03/98

VALIDATION

1. Introduction

I would like us to consider each main part of the previous report.

It would be helpful if you could consider not only whether a particular feature looks useful (because it meets a goal or attenuates a problem), but also whether it is likely to have additional, unpredicted side-effects. I am particularly interested to discover adverse side-effects.

3. Validating the report

3.1 COTS organisation structure (pages 4 to 8)

3.1.1 Are you happy with the new COTS organisational structure?

3.1.2 Would you like to make any changes to the new structure? And if so, why?

3.2 User-problems (page 3)

3.2.1 Are you happy with the six categories of user-problem?

3.2.2 Although "request to borrow manuals", "request for printouts", and "request to buy materials" involve interactions between users and Helpdesk staff, should they be supported in a user-problem management system?

3.2.3 Should there be additional categories or sub-categories?

3.2.4 Are you happy with the concepts of short-term and
long-term user problems and the way that they are defined? For example, depending on the experience of the COTS member of staff, the same user-problem might be classified either way.

3.3 Logging user-problems (pages 9 to 16)

3.3.1 Are you happy with the idea of a single, global integrated computer-supported database for recording user-problems?

3.3.2 Page 14 discusses the kind of user-problem data that would be recorded. Is it necessary to record all this information? Is there additional data that should be recorded?

3.3.3 How should the "problem description" data be structured?

3.4 Users communicating with COTS

3.4.1 Are you happy with the idea of publishing relevant details of COTS staff (see page 21)?

3.5 COTS communicating with users

3.5.1 Are the three categories of communication ok?

3.6 Solving user-problems with the CBPMS

3.6.1 Should this facility be offered to staff users?

3.6.2 Should this facility be offered to student users?

3.7 Prioritizing user-problems

3.7.1 Are the priority categories ok? Should there be fewer? More? Alternatives?

3.7.2 Is the suggested mechanism for increasing the priority of a user-problem ok?
Minutes

Minutes of the meeting of the stakeholders for the user-problem management system held on 18/03/98:

Present: Stewart Green, Heidi Kimber, Rodney Max-lino, Stephen Mullen, Jin Sa, Adam Saunders,

Apologies: Rick Alsopp, Philip Naylor, Sarah Stone

1. Introduction

Everyone was welcomed to the meeting. Relevant introductions were made.

2. Resolving conflict

2.1 Recording short-term user problems

It was agreed that short-term user-problems would not be logged on the computer-based user-problem management system (CBPMS).

2.2 Modes of communication between users and COTS

Rodney Max-lino, the new client for this project, indicated that staff and student users would be treated identically.

He also indicated that details of COTS staff expertise would be published. It is expected that this will help users to contact the right member of staff when they have problems.

It was expected that users would normally contact the Helpdesk in face-to-face mode when first reporting a problem. However, the idea of instituting hard-and-fast rules that would constrain users to just this communication mode for initial communication was rejected. Instead it was expected that choice of channel and target of communication would depend upon the circumstances; users would be expected to use their common sense to make the right decision.

3. Validating the report
3.1 COTS organisation structure

The organisation structures depicted in the previous report (figures one and two) are correct. However, the lines representing "allowed access" are incorrect, and should be removed.

ACTION: SG to modify diagrams.

3.2 User-problems

It was agreed that a CBPMS would not support the following three categories: "request to borrow manuals", "request for printouts", "request to buy materials".

ACTION: SG to modify the report.

Work will be carried out to elaborate the remaining three categories of user-problems.

ACTION: SG to elaborate user-problems.

3.3 Logging user-problems

The requirement for a CBPMS based upon a single, global, integrated database was accepted.

It was agreed that all the information identified in the previous report as needing recording did in fact need recording.

It was agreed that information concerning "estimated time to solution" would be recorded alongside other long-term user-problem data.

No suggestions for further structuring of "problem description" data were put forward.

However, it was recognised that CBPMS users would want to query stored solution data to help them to solve their user-problems. Therefor there is a strong requirement that this data be stored in a way that facilitates such querying.

3.4 Users communicating with COTS

It was agreed that it was a good idea to publish relevant details of COTS staff expertise.

However, it was agreed that no data will be
published on "preferred times" of contact.

3.5 COTS communicating with users

It was agreed that the three categories of communication between COTS staff and users (respond to initial problem report, inform users of significant milestones, inform users of problem solutions) were sufficient.

3.6 Solving user-problems with the CBPMS

It was agreed that the facility to query solution data in a CBPMS would be made available to the users only after this facility was judged by COTS to be both working correctly and presented in the right form.

3.7 Prioritising user-problems

It was agreed that a CBPMS would support the prioritising of user-problems. However, the priority categories that were described in the previous report were not accepted. Max agreed to produce a list of problem category names and their associated definitions.

ACTION: RM to produce a list of priority definitions.

It was agreed that the priority of some user-problems would be automatically increased after a predetermined period of time had elapsed. Increases in priority would continue to take place on a regular basis until a user-problem had been made top-priority.

4. Futures

4.1 Short-term

Publish the minutes of the meeting. (SG: 1 week)

Carry out actions documented in the minutes. (SG and RM: 2 weeks?)

4.2 Medium-term

Elicit stakeholders' requirements for the details of all the reports that they would like to be able to produce from the CBPMS. (SG: 4 weeks)
4.3 Long-term

Produce a detailed requirements document for the CBPMS.
(SG: 8 weeks)

5. AOB