Incorporating Design Explanation

within

Formal Object-Oriented Method (FOOM)

by

Lemai Nguyen

BCs (Honours)

Submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

Deakin University

May 2000
Abstract

Requirements Engineering is a commencing phase in the development of either software applications or information systems. It is concerned with understanding and specifying the customer’s requirements of the system to be delivered. Throughout the literature, this is agreed to be one of the most crucial and, unfortunately, problematic phases in development. Despite the diversity of research directions, approaches and methods, the question of process understanding and management is still limited.

Among contemporary approaches to the improvement of the current practice of Requirements Engineering, Formal Object-Oriented Method (FOOM) has been introduced as a new promising solution. The FOOM approach to requirements engineering is based on a synthesis of socio-organisational theory, the object-oriented approach, and mathematical formal specification. The entire FOOM specification process is evolutionary and involves a large volume of changes in requirements. During this process, requirements evolve through various forms of informal, semi-formal, and formal while maintaining a semantic link between these forms and, most importantly, conforming to the customer’s requirements. A deep understanding of the complexity of the requirements model and its dynamics is critical in improving requirements engineering process management.

This thesis investigates the benefits of documenting both the evolution of the requirements model and the rationale for that evolution. Design explanation explains and justifies the deliberations of, and decisions made during, the design activity. In this thesis, design explanation is used to describe the requirements engineering process in or-
der to improve understandability of, and traceability within, the evolving requirements specification. The design explanation recorded during this research project is also useful in assisting the researcher in gaining insights into the creativity and opportunistic characteristics of the requirements engineering process.

This thesis offers an interpretive investigation into incorporating design explanation within FOOM in order to extend and advantage the method. The researcher’s interpretation and analysis of collected data highlight an insight-driven and opportunistic process rather than a strictly and systematically predefined one. In fact, the process was not smoothly evolutionary, but involved occasional “crisis” points at which the model was reconceptualised, simplified and restructured. Therefore, contributions of the thesis lie not only in an effective incorporation of design explanation within FOOM, but also a deep understanding of the dynamic process of requirements engineering. The new understanding of the complexity of the requirements model and its dynamics suggests new directions for future research and forms a basis for a new approach to process management.
Declarations

I declare that the work presented in this thesis is the result of my own research, except where otherwise acknowledged.

Lemai Nguyen

Melbourne, May 2000
I declare that this thesis in whole or in part has not been submitted for an award, including a higher degree, to any other university or institution.

Lemai Nguyen

Melbourne, May 2000
Acknowledgement

I would like to thank my supervisors without whom I would not have been able to complete this thesis.

I am deeply indebted to my principal supervisor, Professor Paul Swatman, for his guidance, supervision, help and advice throughout my Ph.D. research. To me, he has been not only a supervisor, but also a great mentor, and I can never express my gratitude enough for his insight, encouragement and inspiration.

I am also very grateful to my associate supervisor, Associate Professor Graeme Shanks, for his continued generous support, guidance and patience through my studies.

I would also like to thank many people who provided me with their help and support during my research:

• Researchers and staff of the former-CISR (Centre for Information Systems Research, Swinburne University of Technology), where the a major part of this research project was conducted. I owe special thanks to various researchers and research students of the OO-RE (Object-Oriented Requirements Engineering) group for their valuable contributions and feedback for the Requirements Engineering project used for this research and the data collection and initial data analysis phases of this research.

• Researchers and participants of the Requirements Engineering research group (School of Management Information Systems, Deakin University) for their stimulating discussions and constructive comments during the final data analysis phase of my research.
• Ms Elaine Race for her professionalism, expertise and effort in the editing and proof reading of this thesis. I can never thank her enough for her tremendous support, advice, encouragement, patience, friendship and stimulating discussions.

• Professor Paula Swatman for her kindness and encouragement, especially, in my early years of study.

• My colleagues and friends, Dr. Danielle Fowler, Dr. Craig Parker and Terence Tan for their friendship, kindness, and support.

• Professor Tanya Castleman, Associate Professor Ross Smith and numerous discussants of the action research mailing list (arlist-l@scu.edu.au) for their constructive comments on research methodologies.

• The various anonymous referees who have reviewed parts of this thesis prior to publication in journals and conference proceedings and whose constructive comments have contributed to the clarification and presentation of many ideas and to the quality of this thesis.

I also need to say thanks to many other people for their kind assistance. Although I cannot thank everyone individually here, I sincerely thank each and everyone of them.

Last but not least, I would also like to thank my parents, my brother, my daughter and my (extended) family for all their love, support and encouragement throughout the years. I am especially in debt to my parents, who have been teaching me the value of education and passing onto me their love and devotion to science from my very early years of childhood. Especially, I wish to express my love and thanks to my husband, Tony Ngo, for his continuing love, patience, support and understanding.
Contents

Abstract ........................................ i

Acknowledgement .............................. iii

Notational Conventions ........................... xvi

1 Introduction .................................. 1

1.1 Research interests .......................... 1

1.2 Research objectives ........................ 3

1.3 Research setting ........................... 3

1.4 Structure of the thesis ........................ 4

2 Literature Review ........................... 7

2.1 Requirements Engineering ..................... 8

2.1.1 Definitions ................................ 8

2.1.2 Requirements engineering as a complex problem understanding
and solving activity ............................ 13
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.3</td>
<td>Characteristics of problems in requirements engineering</td>
<td>16</td>
</tr>
<tr>
<td>2.1.4</td>
<td>Requirements engineering process</td>
<td>21</td>
</tr>
<tr>
<td>2.1.5</td>
<td>The volatility of requirements</td>
<td>33</td>
</tr>
<tr>
<td>2.1.6</td>
<td>Requirements engineering as a social and communicative process</td>
<td>40</td>
</tr>
<tr>
<td>2.1.7</td>
<td>Discussion</td>
<td>46</td>
</tr>
<tr>
<td>2.2</td>
<td>Design Explanation</td>
<td>50</td>
</tr>
<tr>
<td>2.2.1</td>
<td>What is design explanation?</td>
<td>50</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Approaches to design explanation</td>
<td>57</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Issue-Based Information Systems (IBIS) and its descendants</td>
<td>59</td>
</tr>
<tr>
<td>2.2.4</td>
<td>gIBIS and itIBIS</td>
<td>61</td>
</tr>
<tr>
<td>2.2.5</td>
<td>PHI</td>
<td>70</td>
</tr>
<tr>
<td>2.2.6</td>
<td>Design Space Analysis with QOC</td>
<td>74</td>
</tr>
<tr>
<td>2.2.7</td>
<td>Other design explanation notations</td>
<td>94</td>
</tr>
<tr>
<td>2.2.8</td>
<td>Summary and conclusion</td>
<td>110</td>
</tr>
<tr>
<td>2.3</td>
<td>FOOM</td>
<td>111</td>
</tr>
<tr>
<td>2.3.1</td>
<td>An introduction to Formal Object-Oriented Method</td>
<td>112</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Why FOOM?</td>
<td>114</td>
</tr>
<tr>
<td>2.4</td>
<td>Synthesising views</td>
<td>116</td>
</tr>
<tr>
<td>3</td>
<td>Research Methodology and Research Design</td>
<td>118</td>
</tr>
</tbody>
</table>
3.1 Research objectives ........................................ 118

3.2 Research approaches in Requirements Engineering ............ 121
   3.2.1 The Information Systems and Software Engineering disciplines . 121
   3.2.2 Research paradigms in Information Systems: Positivism and Inter-
       pretivism ............................................ 122
   3.2.3 Choosing an appropriate research approach for the project ... 134
   3.2.4 Possible research approaches ............................ 135

3.3 Justification of research approach and research design .......... 140
   3.3.1 A programme to develop FOOM — the research context .... 140
   3.3.2 Justification of action research as a valid choice .......... 141
   3.3.3 Research design ................................... 143

4 Using ad-hoc Design Explanation within the Requirements Engineer-
   ing Process ........................................... 153

   4.1 Research process: observation and data collection ............. 154
      4.1.1 Choosing a process-oriented design explanation notation .... 154
      4.1.2 The requirements engineering project ..................... 156
      4.1.3 Using IBIS within the FOOM process ..................... 157

   4.2 Reflection on the use of IBIS ................................ 159
      4.2.1 A summary of findings .............................. 159
4.2.2 An analysis of findings ........................................ 160

4.3 Using IBIS more effectively ...................................... 176

4.3.1 A summary of findings ........................................ 176

4.3.2 An analysis of findings ........................................ 176

4.4 Conclusion .......................................................... 179

5 Supplementing ad hoc IBIS with post hoc QOC .................. 181

5.1 Research process: observation and data collection ............... 183

5.1.1 Choosing a process-oriented design explanation notation .... 183

5.1.2 The requirements project—an extended application ............ 186

5.1.3 Using QOC and IBIS within the FOOM process ............... 187

5.2 The usefulness of IBIS and QOC .................................. 191

5.2.1 A summary of findings ........................................ 191

5.2.2 An analysis of findings ........................................ 192

5.3 How to use IBIS and QOC within FOOM effectively ............ 218

5.3.1 A summary of findings ........................................ 218

5.3.2 An analysis of findings ........................................ 219

5.4 Conclusion .......................................................... 228

6 New Understandings of the Requirements Engineering Process .... 230

6.1 The catastrophe-cycle requirements modelling process .......... 233
6.2 Complexity of the requirements model ........................................... 239

6.2.1 Building up an initial understanding of the complexity of the
requirements model ............................................................................. 239

6.2.2 Building up a catastrophe-cyclic requirement modelling process . 254

6.3 Supplementing IBIS with QOC—Increasing essential knowledge .... 258

6.4 Summary ....................................................................................... 267

7 Conclusions and Future Research ......................................................... 269

7.1 Summary ....................................................................................... 269

7.2 Summary of findings ....................................................................... 270

7.2.1 Principal research question 1 ...................................................... 271

7.2.2 Principal research question 2 ...................................................... 272

7.2.3 New understanding of the requirements engineering process and
its dynamics ....................................................................................... 275

7.3 Implications .................................................................................... 277

7.3.1 A new approach to using design explanation within FOOM .... 277

7.3.2 Requirements engineering ........................................................... 281

7.3.3 Design explanation ...................................................................... 284

7.3.4 How should we train requirements engineers? .............................. 288

7.4 Future research .............................................................................. 288
# List of Figures

2.1 IBIS model .................................. 60

2.2 gIBIS model (Conklin and Begeman, 1988) ................. 62

2.3 An extension to IBIS (Pries-Heje, 1993) .................. 67

2.4 REMAP (Ramesh and Dhar, 1992) ....................... 69

2.5 Example of a PHI map (Fischer et al., 1991) ............... 71

2.6 QOC model (MacLean et al., 1991) ........................ 75

2.7 Locating IBIS and QOC within a two dimensional space (Extracted from Shum, 1991b, page 263) ...................... 84

2.8 Example of the Toulmin notation (Toulmin et al., 1984) ... 96

2.9 Generic model for recording design deliberation and artefact (Potts and Bruns, 1988) .......................... 98

2.10 Inquiry Cycle model (Potts et al., 1994) .................. 99

2.11 Abstraction levels in EOM (Moreno and Souveyet, 1993) 104

2.12 Basic model of development loop (Moreno and Souveyet, 1993) 105
5.6 Three possible positions for Issue B .................. 200
5.7 A QOC analysis example .................................... 205
5.8 Overview of the literate specification technique (Johnson, 1996) ...... 211
5.9 FOOM and Design Explanation—Collaborative in literate specification . 211
5.10 Using IBIS with and without QOC supplement within requirements engineering—the process ........................................ 214
5.11 Using IBIS with and without QOC supplement within requirements engineering—the searching time ........................... 217
5.12 Process of documenting evolution of requirements specifications and the rationale for the evolution ........................................ 221
6.1 Qualitative interpretation of expected FOOM process .................. 234
6.2 The evolution of the complexity of FOOM models ..................... 234
6.3 Evolution of requirements for the CASE tool (extracted from intermediate specifications) ........................................ 242
6.4 Improved classification as a result of insight (extracted from intermediate specifications) ........................................ 251
6.5 A preliminary pattern of the requirements evolution ..................... 254
6.6 A qualitative explanation of the essential and incidental complexity in requirements models ........................................ 255
6.7 Catastrophe-cycle requirements modelling process ..................... 256

xiii
6.8 QOC analysis created to support the restructuring of the model—the first example ........................................... 260

6.9 Different ways of modelling a graph .............................................. 262

6.10 QOC analysis created to support the restructuring of the model—the second example ........................................... 263

6.11 A linear stream of IBIS arguments is formed. Reproduced from Figure 4.3 for readers’ convenience ........................................... 264

6.12 Using QOC to review IBIS Issues ........................................... 265

6.13 Gaining the essential complexity using QOC ........................................... 265

6.14 Using both IBIS and QOC within FOOM ........................................... 266
List of Tables

5.1 Differences between IBIS and QOC collected data 226
Notational Conventions

Throughout this thesis, the following conventions are used:

Sans Serif

Used for components of design explanation notations. For example:

Issue, Criterion, Objects-to...

Italic

Used for naming modelling elements, primarily FOOM modelling objects and classes in Chapters 4, 5 and 6. For example:

Node, Link, Project...

Also used for other general purposes, such as emphasis and direct quotations.

Bold

Used for describing action research in terms of an intellectual framework, method, application (see Chapter 3). For example,

F, M, A...

Also used for other general purposes, such as emphasis and dividing the text.
Chapter 1

Introduction

1.1 Research interests

There is growing recognition within the information systems and software development industry as well as the research community that requirements engineering is a crucial phase of the Systems Development Life Cycle (see for example Boehm, 1976; Davis, 1990). Clearly, the requirements engineering phase needs to be supported and well-managed in terms of both product and process.

The requirement engineering process is a process of understanding and specifying the customer’s requirements of the system to be delivered. The end product of requirements engineering is a precise specification to which the system is expected to conform. Both the requirements specification and the requirements engineering process need to be well understood, supported, monitored and controlled.

This thesis sets out to explore and investigate a new way of improving understandability of the requirements specification and of managing the requirements engineering
process. The exploration and investigation was based on the following areas of research in systems development:

**Requirements engineering** This is an area of research which addresses the perspectives and issues associated with specifying the client’s requirements as well as understanding and improving the process of doing so. The primary interests of this thesis include issues related to the evolution of the requirements specification/model and understanding the dynamics of this evolutionary process.

**Design explanation** This is an area of research which addresses the issues related to the representation and explanation of the rationale behind the human decision making activities, such as legal decision making, policy making, planning, architecture, and systems design\(^1\). The general hypothesis of the thesis is design explanation can be beneficial and therefore can be incorporated within requirements engineering.

**Formal Object-Oriented Method (FOOM)** FOOM is a contemporary requirements engineering approach. It is based on a synthesis of socio-organisational theory, the object-oriented approach, and mathematical formal specification. FOOM was chosen as a specific requirements engineering method, which allows the researcher to gain insight for the investigation through the application of the method.

---

\(^1\)Design explanation is often referred to as design rationale in the literature, the term “design explanation” is deliberately chosen for use in this thesis. This will be explained in Chapter 2.
1.2 Research objectives

The thesis is concerned with improving both the product and process in requirements engineering using design explanation. Since the understandability of the product and support for the process of requirements engineering were seen as primary, design explanation which represents and explains the deliberations underlying the design process seemed to provide potential benefits to requirements engineering. Therefore, this thesis was designed to explore and investigate:

- Whether or not design explanation can be useful to understand the product and to support the process of requirements engineering?

- If so, how might design explanation be incorporated systematically within requirements engineering?

The expected research outcome provides answers to the above questions and, perhaps, a systematic approach to using design explanation within FOOM.

1.3 Research setting

Action research was adopted for the exploration and investigation of the above research questions. Action research’s hermeneutics cycles allowed the research ideas/concepts to be generated, refined and evolve. The active and reflective characteristics of action research enabled the researcher to:

- explore and investigate the incorporation of design explanation within FOOM

- experience the development process and reflect upon it herself
focus on her observation and action during of the development process at the conceptual level

interpret the story of what was actually happening during the project, not relying on retrospective interviews or the accuracy of research subjects’ explanation of what they did, and how and why they did it.

In addition, it was decided that the research be carried out in a real requirements engineering project with a university environment in Australia, in order to allow a flexible control of the research process to pursue research interests and theory development, while maintaining the practical values of research outcome.

The action research involved two hermeneutic cycles which aimed at developing a qualitative explanation of the application of design explanation in the context of the requirements evolutionary process. Both the application of design explanation and the process of requirements evolution were foci of the study.

1.4 Structure of the thesis

The thesis reports this action research as follows:

Chapter 2 This chapter critically reviews the literature in requirements engineering and design explanation and introduces FOOM. The this chapter aims at:

- synthesising conclusions from the three themes and developing a framework of ideas, which would serve as an intellectual lens for interpreting the phenomenon and defining the findings of the study
- providing a strong rationale for the research
• justifying the choice of FOOM as a specific requirements engineering method for use in the action research

Chapter 3 This chapter describes and justifies the research approach and design. Firstly, the chapter formulises the research questions. Secondly, it discusses a range of research approaches in Requirements Engineering and a number of issues considered when choosing an appropriate research approach for the project. Finally, it justifies the chosen research approach (action research) and presents the research setting and research design.

Chapter 4 This chapter describes the first research cycle which involves an application of an *ad hoc* design explanation notation (Issue Based Information System—IBIS) within the FOOM process. It is presented in terms of research process and findings. The analysis focuses on the primary research questions. The findings include both benefits and limitations of using IBIS within FOOM and suggestions how to use IBIS effectively within FOOM. An initial understanding of the requirements engineering process is also gained.

Chapter 5 This chapter describes the second research cycle which involves an investigation with a view to supplementing IBIS with a *post hoc* design explanation notation (Question Option Criteria—QOC). It is presented in terms of research process and findings. The findings and questions raised from the first research cycle are analysed and refined. The analysis focuses on the primary research questions. The findings present the benefits of using both IBIS and QOC including the benefits of supplementing IBIS with QOC and suggestions of how to use these notations effectively within FOOM.

Chapter 6 The findings from the two research cycles lead to a *post hoc* examination
of the requirements engineering process in detail and the application of both the IBIS and QOC notations from the process perspective. This chapter describes this examination and presents a new significant understanding of the requirements engineering process, its dynamics and its creative and opportunistic characteristics. The usefulness of IBIS and QOC is analysed in the context of this new understanding.

Chapter 7 This chapter systematically summarises the research findings analysed in Chapters 4, 5 and 6, discusses implications of these findings based on conclusions reached in the literature review (Chapter 2), and outlines directions for further work.

Appendix A An introduction of FOOM by Swatman (1996). This introduction is included here in order to assists readers who are not familiar with FOOM. A better understanding of FOOM is useful but not essential in following the thesis.

List of publications This is a list of publications which flow from this thesis

In general, this research offers contributions to knowledge in respect of both requirements engineering and design explanation. In particular, the research leads to a new systematic approach to incorporating design explanation within FOOM.
Chapter 2

Literature Review

This chapter presents and develops three themes, requirements engineering, design explanation and FOOM.

First, the chapter elaborates an understanding of the requirements engineering process. In essence, this section contributes towards drawing a framework of ideas within which the findings from the research will be defined.

Second, the chapter reviews existing approaches and studies to design explanation. Design explanation focuses on representing and explaining the rationale behind the design of an artefact.

Third, the chapter introduces FOOM, a requirements engineering method chosen for the research project. FOOM is presented in terms of the method’s theoretical bases, structure and process. The choice of FOOM is explained.

Finally, the chapter consolidates the major conclusions reached in each of the three themes and provides a strengthened motivation for the research objectives (introduced in Chapter 1 and formally defined in Chapter 3).
This chapter is not intended to be a full treatise about requirements engineering, design explanation or FOOM, rather it includes only essential concepts which underpin the research reported in this thesis.

2.1 Requirements Engineering

2.1.1 Definitions

One of the most significant topics of present debate in requirements engineering is the quality of requirements specification and the requirements process. Despite the diversity of research directions, approaches and methods, the question of process understanding and management is still limited. In this chapter, I will critically review the literature, discuss topics of current debate, expose shortcomings of research to date and draw up conclusions providing a strong rationale for the research purposes.

I begin by defining Requirements Engineering for the purposes of this thesis. It is important to do so because:

- the term requirements engineering has a variety of connotations for the various researchers and practitioners who use the term; and, to confirm the matter further,

- what we term requirements engineering has been labeled variously by other researchers and practitioners.

A formal definition is the first step in the process of linking the numerous streams of relevant literature so as to provide a description of the communities understanding which forms a foundation for the work reported here.
“Requirements engineering can be defined as the systematic process of developing requirements through an iterative co-operative process of analysing a problem, documenting the resulting observations in a variety of representation formats, and checking the accuracy of the understanding gained.”

(Loucopoulos and Karakostas, 1995, page 13)

This definition is widely accepted and particularly discussed by Macaulay (1996). Although this definition is rather simplistic, it emphasises the iterative process of elicitation, representation and validation of information (and software) system requirements. It also recognises the social or communicative aspect of Requirements Engineering which is referred to as a co-operative process.

This definition reflects the deep analysis offered by Pohl (1993) according to which Requirements Engineering can be seen from three dimensions:

**Specification dimension** Requirements are transformed from an initial opaque state to a final complete specification. This dimension deals with cognitive problem of conceptual modelling of requirements. This dimension brings together research streams about the nature of requirements (Kilov and Ross, 1994; Davis, 1993) and different paradigms in data modelling, such as data-oriented approaches, e.g. Entity-Relationship (Chen, 1976; Kent, 1983; Barker, 1989; Elmasri and Navathe, 1989; Simsion, 1994), fact oriented approach (Nijssen and Halpin, 1989), process-oriented approaches, e.g. data flow diagrams, and object-oriented approaches (Wirfs-Brock et al., 1990; Rumbaugh et al., 1991; Jacobson, 1994; Booch, 1994; Henderson-Sellers and Edwards, 1994; Booch et al., 1999; Henderson-Sellers et al., 1998; Firesmith et al., 1998).
**Representation dimension** Requirements can be represented using different formalisms (*i.e.* meta models). This dimension reflects a growing body of research into requirements representation methods and notations for capturing semantic information. These notations range from informal (*e.g.* natural and user-oriented languages), semi-formal (*e.g.* Entity-Relationship (Simsion, 1994), and Object-Oriented diagrams (Rumbaugh et al., 1991; Jacobson, 1994; Booch, 1994; Booch et al., 1999; Firesmith et al., 1998) to formal languages with rich and well defined semantics (*e.g.* Z (Diller, 1994; Lightfoot, 1991; Wordsworth, 1992) and Object-Z (Duke and Rose, 1995; Smith, 1999)).

**Agreement dimension** There may be disparities and inconsistencies between different views of different groups of involved people, customers, end-users, managers, requirements engineers, developers/implementors, and so forth. This dimension deals with communicative and social activities among these people to support the resolution and integration of different viewpoints (Checkland and Scholes, 1990; Lewis, 1993; Sommerville et al., 1993; Goguen and Linde, 1993).

Having taken into account the above, this thesis particularly emphasises the two following aspects of Requirements Engineering:

- Requirements engineering as an iterative process of identifying and developing requirements, and managing the communication between different groups of involved people in order to produce a requirements specification.

- Requirements engineering as a dynamic process of change, during which requirements evolve through various representations and are transformed from initial opaque statements to a precise specification.
In recent years, many approaches, taxonomies, methods and techniques have been proposed to assist the requirements engineering task. A wide range of requirements approaches are described and critically reviewed in the literature (Macaulay, 1996; Loucopoulos and Karakostas, 1995; Ilivari et al., 1998; Pohl, 1993; Macaulay et al., 1990; Christel and Kang, 1992; Wood-Harper and Fitzgerald, 1982; Bickerton and Siddiqi, 1993; Hirschheim and Klein, 1989). Requirements engineering approaches are diverse. Rather than simply being alternatives, they seek to do or to achieve different things (Wood-Harper and Fitzgerald, 1982). They may reflect different fundamental philosophical assumptions (Hirschheim and Klein, 1989; Bickerton and Siddiqi, 1993; Wood-Harper and Fitzgerald, 1982; Ilivari et al., 1998); they may seek to achieve different purposes including the provision of a better problem understanding or support for problem solving (Wood-Harper and Fitzgerald, 1982); they may be data-oriented, function-oriented, object-oriented or control-oriented (Olle et al., 1991; Avison and Fitzgerald, 1995; Simsion, 1994; Sommerville, 1996; Dorfman, 1997); and they may focus on different aspects ranging from socio-organisational to technical, or a combination of both (Checkland and Scholes, 1990; Henderson-Sellers and Edwards, 1994; Siddiqi et al., 1994; SWATMAN, 1996). In the light of the above discussion of Requirements Engineering, it appears that current approaches tend to emphasise the elicitation, representation, and social and communicative issues. The research described above relates to requirements engineering in the first of the aspects above. The focus is on deriving a desirable and high quality specification which is the end product of requirements engineering.

In general, most requirements engineering approaches fall short in providing adequate support for the process of requirements specification. Although this aspect is highlighted
and addressed in a number of previous studies, for example (Curtis et al., 1988; Guin-
don, 1990b; Visser, 1994a; Ramesh and Dhar, 1992; Ramesh and Edwards, 1993; Som-
merville and Sawyer, 1997), either focus is limited to psychological cognitive behaviour
of the requirements engineer at the individual level; or the process is seen as a systematic
and incremental activity. Consequently, the rudimentarily handling of the traceability
and configuration management issues is considered in relation to techniques/methods
for the improvement of the process. Moreover, by presupposing an insufficient or im-
proper understanding of the requirements engineer’s activities and the requirements
process, the methodology might handicap the requirements engineer rather than sup-
port him/her. Undoubtedly, there is a need for a systematic approach which is grounded
on a deep understanding of the whole process. The research reported in this thesis is
undertaken with a view to gaining a deep understanding of the requirements engi-
neering process and investigates a systematic approach to explaining and supporting
the process (of requirements evolution); therefore, improving both the requirements
engineering process and the understandability of requirements specifications.

In the discussion below, the literature in Requirements Engineering will be reviewed in
terms of the cognitive task, the characteristics of requirements problems, the process,
the volatility of requirements, and the socio-communicative issue. Conclusions derived
from each discussion will explain motivations for the research objectives and build up
an intellectual framework of ideas within which learning will be defined.
2.1.2 Requirements engineering as a complex problem understanding and solving activity

Requirements engineering, similar to design, is a problem solving activity. This claim is commonly agreed upon throughout the literature. According to Malhotra et al. (1980, page 120) “problem solving occurs in moving from a problem state to a non-problem state” and often involves a number of transformations. Defining and exploring a problem situation, and transforming informal, incomplete and ambiguous requirements into a precise specification, is a complex, cognitive activity, in which the requirements engineer “is forced to engage in both broader conceptual thinking, as well as focused problem-solving activities” (Batra and Davis, 1992, page 87).

In fact, requirements engineering should be described more completely as a problem understanding and solving activity. Research in the 1980s tends to treat the development of the solution as being “preceded by an independent and complete phase of problem understanding”. (Guindon, 1989, page 729). However, analysing protocols recorded from observing various designers performing a design task within two hours, Guindon (1989) describes this as a knowledge discovery process and that problem understanding is not a fully completed process that precedes design. Her study reveals that this process involves the decrease of incompleteness and ambiguity in the emergent model of requirements. In other words, this process involves understanding and elaborating requirements with a purpose of decreasing “the incompleteness and ambiguity of the informal specification” (Guindon, 1989, page730). She suggests that CASE tools may enable the editing and organisation of design issues and decisions and thus support this process of knowledge discovery.

Further, Visser (1992, p.92) advises that requirements engineers are not given fixed
or defined problems—that in fact, they construct them. Visser and Falzon’s (1988) observations also lead to the conclusion that the problem definition does not precede the design process, it may be provided by the designer or the client later, the designer may keep or modify the problem constraints during the process. Observing an expert technician with 30 years of experience designing a new type of antenna over nine weeks, Visser (1992, page 100) finds that each intermediary solution constitutes both “a further specification of the problem” and “a new problem to be solved”. He considers the problem solving action as consisting of three steps: “problem-representation construction, solution development and solution evaluation”.

Requirements engineering is a cognitive design process, it involves the conscious mental effort to come to a decision. A number of cognitive research studies into similarities and differences between novices and expert designers generate many insights and develop an understanding of this cognitive process of problem understanding and solving. These studies find that both experts and novices share similarities in cognitive behaviour and dependency relationships between behaviours.

Sutcliffe and Maiden (1992) investigate the cognitive design process of 13 novice analysts through a 35 minute empirical experiment. Analysing verbal protocols and retrospective questioning from the development of a specification for a delivery scheduling system, Sutcliffe and Maiden (1992) categorise and model the mental behaviour as consisting of complex dependencies between Information-gathering, Assertions, Conceptual modelling, Planning, Recognising goals and Reasoning. According to the authors, the strongest associations are between Information-gathering, Assertions and Conceptual modelling. These associations are explained as a representation of the analytical side of understanding the problem domain. According to their observation, most of the sub-
jects tend to use a Hypothesis (solution) -Generate, -Develop and -Test mode to solve requirements problems. After being tested, hypotheses can be discarded or modified.

Batra and Davis’s (1992) empirical study examines similarities and differences between novice and expert data modellers. This study shows the process model of conceptual data modelling as consisting of three levels. The Enterprise level focuses on developing an understanding of the problem domain. The Recognition level focuses on understanding a sub-problem at-hand and mapping the appropriate knowledge from the requirements engineers’ experience. The Representation level involves both representation of requirements and verification of that representation. Although this classification is rather simplistic, it explicitly shows that understanding is a part of the requirements engineering process. According to the findings, novices spent most of the time at the Representation level (50%) and least time at the Enterprise understanding level (20%). However, experts spent equal time at these levels (42-43%). Novices focus on structuring requirements while experts’ efforts were directed towards developing a holistic understanding of the problem, abstracting, categorising and representing information. Batra and Davis (1992) also find that experts cycle through level.

Building upon Batra and Davis’s (1992) study, Chaiyasut and Shanks (1994) study differences between four experts and four novices in a conceptual data modelling task. They take a view more focused on the data perspective. The authors categorise the cognitive process into six detailed types of Understanding, Recognising Goals, Planning, Recognising and Reusing Experience, Searching Solutions and Representing Information. The authors find that novices spend most of their time in understanding the problem while the experts spent most of their time on the modelling task. Novices’ models are developed “literally” from the problem description while experts’ models
contain “many concepts not explicitly mentioned in the case problem” (Chaiyasut and Shanks, 1994, page 320). Models developed by experts are more comprehensive, complete and hold a holistic view of the problem.

Although the cognitive behaviours in design (or data modelling) are classified and analysed differently through these studies, a common conclusion is that requirements engineering involves a traversal between problem understanding and solution development processes. The requirements engineer’s understanding of the problem increases as the incompleteness and ambiguity of the problem are reduced. However, little is said about how the requirements engineer’s knowledge of the problem space is gained and increases in the emergent model of requirements. There is a need for the further investigation of the dynamics of requirements models and of ways to monitor and support this process of knowledge discovery.

In addition, most previous empirical research in the area observes analysts/designers working on small cases over short periods of time (often several hours) and relies on the accuracy of verbal protocols and memory by research subjects in retrospective interviews. The generalisation of conclusions to the complex reality of real world professional practice is not convincing. This thesis has more closely approximated reality by focusing on realistically scaled problems over a long period of time. The requirements engineering process under study is observed and documented. The dynamics of the requirements model are recorded and analysed in detailed.

### 2.1.3 Characteristics of problems in requirements engineering

The problems presenting in requirements engineering are ill-structured. The ill-structuredness in requirements engineering is defined as the incomplete and ambigu-
ous representation of problems, the multi-discipline domains and knowledge, the non-
deterministic approach to solving requirements problems and the open-ended nature of solutions (Guindon, 1990b; Carroll et al., 1979; Batra and Davis, 1992).

It is apparent that user requirements, expressed in natural language, are often vague, incomplete, and may be contradictory (Kilov and Ross, 1994; Lightfoot, 1991; Christel and Kang, 1992). Throughout the requirements development process, as the problem space is explored, requirements continue to be acquired, clarified, refined, and modelled by the requirements engineer. The requirements engineer learns about the client’s organisation, current practice and application domain. At the same time, the client is “educated” (by the requirements engineer) about the technical knowledge, including incompleteness, feasibility and possible options for requirements. In addition, the clients may not know what they want, and they may change their preferences over time (Wordsworth, 1992). Furthermore, the understanding and analysing requirements often involves subjective interpretation and perception by different participants. During the process, the client has to make decisions on conflicts identified by the requirements engineer (Fowler and Swatman, 1997). In other words, these problems are intrinsic to the client’s requirements and often require both the requirements engineer and the client to engage in mutual learning activities in order to understand and specify the client’s requirements (see for example, Floyd et al., 1989; Kyng, 1991; Carmel et al., 1993; Anderson and Crocca, 1993; Macaulay, 1996; Simonsen and Kensing, 1993; Carroll and Swatman, 1998).

The multi-disciplinary domain of requirements engineering makes the problem space rather broad and undetermined. According to Batra and Davis (1992, page 87), requirements are constructed in a rather “open-ended” and “semantically rich problem
space”. Guindon (1990b) argues that the design process is characterised by the integration and coordination of multiple domains of knowledge. Indeed, the field study by Curtis et al. (1988) confirms that the knowledge of the problem domain is critical to the success of a project. Simsion (1994) also recognises the importance of the knowledge of the problem domain in addition to the conceptual modelling technique.

The above factors contribute to the non-deterministic nature of the requirements engineering process. Indeed, there is no predetermined way to transform the informal requirements, which are rather incomplete, ambiguous and perhaps contradictory, into a precise specification, expressed in formal and/or semiformal notations. This is explained by Carroll et al. (1979) as follows. “There is no problem tree representation for these problems” because requirements “are too complex; and there are many ways to ‘solve’ them” (Carroll et al., 1979, page 84). The method used in requirements modelling can be seen as a combination of retrieving and modifying “standard” structure or generic models from accumulated experiences and/or creating new models which are most satisfying for the current case. Potts (1989) in his study of systematic approaches to representing process information sees the design method as a combination of “Genericity” and “Customisation”. It is agreed throughout the literature that requirements engineering requires creativity and heuristics as well as “standard” modelling techniques (Guindon, 1990b; Khushalani, 1997; Robillard, 1999). In particular, requirements modelling often involves novel uses of techniques and approaches when dealing with unfamiliar systems and/or domains (Guindon, 1990b). While most authors agree on the knowledge intensiveness and creativity in requirements engineering, the literature is limited in describing how requirements engineers actually use their knowledge and creativity during the requirements engineering process.
Human creative thinking offers a wealth of choices of how to present business rules of a problem domain. Hence, different approaches may lead to one or more possible solutions (requirements models). Moreover, the requirements are usually flexible enough to accommodate a variety of different solutions rather than a unique expected solution. The quality of a solution may be evaluated in respect of general criteria, such as completeness, integration, simplicity (as opposed to complexity), understandability, flexibility and implementability (Simsion, 1994; Moody and Shanks, 1994; Moody and Shanks, 1998; Roseman, 1998). A number of frameworks have been proposed and developed for the evaluation of quality in solution or requirements model (Lindland et al., 1994; Moody and Shanks, 1998; Roseman, 1998). These frameworks guide the measurement of the quality of the requirements model in terms of how well the produced model supports the requirements and enforces the business rules within them.

The recognition that there are a number of workable requirements models rather than a uniquely “best” model presents two obvious questions: Are there other alternatives to be considered? What are the pros and cons of each alternative? Therefore, the knowledge of why a particular specification was chosen among other alternatives is very important in understanding a requirements model. This knowledge is important not only to requirements engineers but also to other stakeholders, such as clients and, in the future, maintainers. Although there is a common agreement on the potential of this knowledge in understanding the requirements model in research and practice (Christel and Kang, 1992; Jarke, 1998; Ramesh, 1998; Shanks and Simsion, 1991), there has been little research into the use of design rationale within requirements engineering in justifying and improving the understandability of the requirements model. Research into capturing the deliberation knowledge behind the design decisions will be reviewed.
In summary, the problems in requirements engineering are characterised as being ill-structured and requiring both intentional and creative processes to solve them. Requirements problems/goals are open-ended and controversial. Requirements engineering is best characterised by “creativity, judgement, and dilemma handling, rather than by objective scientific methods” (Fischer et al., 1996, page 269). Gero (1990, page 28) argues that design is a “goal-oriented, constrained, decision-making, exploration, and learning activity that operates within a context that depends on the designer’s perception”.

My conclusions suggest that the documentation of how a requirements specification is achieved may contribute to a better understanding of the process. This view of requirements engineering also leads to a preference to the descriptive over the prescriptive approach to documenting and managing the process. This perspective agrees with Visser’s (1992, page 92) argument for “design-assistance” as opposed to design-automation.

Having argued that requirements engineering is a complex and creative process with the purpose of producing an artefact (a specification), it is clear that:

- the why underlying the specification is crucial in understanding the requirements model,
- there is a need for an investigation into how the requirements engineer uses his/her knowledge and creativity as this knowledge is important in understanding and monitoring the requirements engineering process.
- because of the non deterministic nature of requirements engineering, a descriptive approach to supporting this activity is preferable.
2.1.4 Requirements engineering process

Throughout the literature, there is considerable variation in the descriptions of the process of requirements engineering. Here, I review previous studies and conclude that the process needs further consideration and that an understanding of the **how** and **why** underlying this process is central to understanding the process.

**Top-down perspective**

Traditionally, the process is seen as hierarchically organised, on the reductionist basis, whereby a problem is decomposed progressively into ever smaller, and eventually manageable units that are resolved and then integrated to form the solution to the problem (Kant and Newell, 1984; Jeffries et al., 1981). Strategies used in requirements engineering are often described as divide-and-conquer or generate-and-test and are claimed to be domain-independent. Observing design processes by both experts and novices, Jeffries et al. (1981, page 279) find that “The decomposition process is central to the successful derivation of a software design. It serves to break a problem down into manageable and minimally interacting components...For experts, the decomposition and subproblem selection processes of the design schema dictate the global organisation of their design behaviour.”. These authors believe that experts tend to apply a breadth-first technique to explore many subproblems at the same level of abstraction before moving into depth (or successive refinement of details) while novices tend to adopt a depth-first approach by expanding details of a subproblem. Although deviations from the top-down design were observed, such as starting the decomposition in the middle of the hierarchy or processing only some branches of the hierarchy, they are explained as exceptions when the designer finds a known solution or faces specially difficult situations. However, the authors themselves do not explore the inferences between problem areas and
solutions/subsolutions during the design process (Guindon, 1990b). In addition, the authors simplistically view other techniques, such as bottom-up, middle-out, inside-out as alternatives when the top-down is not suitable. They criticise the literature as not providing a map between types of problems and suitable design strategies.

Adelson et al. (1984) and Adelson and Soloway (1985) present the balanced development design process. They study novice and expert designers designing an electronic mail system, a library record keeping system and an interrupt handler with which they had differing degrees of familiarity. According to the authors, the design “model started out at an abstraction level ... and progressed to a concrete one” and that “only one level of representation is focused at a time” (Adelson and Soloway, 1985, page 1354). The authors also observe that the experts simulated the mental model to integrate familiar structures in novel ways and test their partially completed designs. They speculate that balanced development allowed the expert designers to run simulation smoothly. Adelson et al. (1984) introduce the notion of demonds which are the designer’s notes. Note taking is regarded to as an important technique experts use to maintain balanced development by allowing them to forget some concerns and reminding them of these concerns at an appropriate level of abstraction. In addition, they also observe that experts allow themselves to violate systematic expansion and detailed exploration of mental model only when dealing with familiar problems (the interrupt handler problem (Adelson and Soloway, 1985)).

Although the above empirical studies examine novices and experts performing design tasks with different degrees of familiarity and requiring integration of different sources of knowledge, most of the case problems under study had well-defined and limited goals: computational geometry (the convex-hull construction) (Kant and Newell, 1984),
book indexing system (Jeffries et al., 1981), library system and electronic mail system (Adelson et al., 1984; Adelson and Soloway, 1985). By staying with such situations, they do not exclude that alternative processes (which may not necessarily be hierarchical process) may happen in ill-structured situations, *i.e.* more complex or commercial projects, especially when communication and negotiation by designers and clients is involved.

**Cyclic and systematically evolutionary perspective**

More recent literature often describes the requirements engineering process as cyclic with systematically evolutionary development of requirements models (Malhotra et al., 1980; Carroll et al., 1979; Thomas and Carroll, 1979; Loucopoulos and Champion, 1989; Christel and Kang, 1992; Loucopoulos and Karakostas, 1995).

There cannot be a complete description of the requirements engineering process as there can be for well-structured problems, however, "the organisation of behaviour is not arbitrary" (Carroll et al., 1979, page 84). Analysing the client designer dialogues recorded from two empirical studies (designing a library system and a schedule for a hypothetical library), Carroll et al. (1979, page 85) find that the process is cyclic with "each (cycle) concerned specifically with a part of the overall design" due to limited memory and attention of the designer. Each cycle consists of the introduction of requirements (a goal/subgoal) by the client, the investigation and suggestion of a solution by the designer, and the validation of the suggested solution by the client. Then the next cycle starts with either a new requirements (another goal/subgoal) or the elaboration of the previous requirement if the proposed solution was rejected. They also find that there are forwards and backwards relationships between cycles. While pointing out that designers do decompose the ill-structured problem into less complex and
more well-defined units, they also acknowledge the difficulty in achieving a coherent neostructuralist analysis of ill-structured problem solving into basic behavioural and cognitive elements.

Malhotra et al. (1980) also make a similar observation in two empirical studies, one on restaurant design and the other on software design. They find that the requirements modelling process is cyclic and each cycle involves three stages of goal elaboration, design generation and design evaluation. According to the authors, the goal of the first stage is to state and analyse user requirements. The output requirements from here are used as input in searching for different solutions in the second stage. During the third stage, these alternative options are continually evaluated by a trade-off analysis using certain preferred criteria. During the evaluation stage some newly appeared (intermediate) questions often trigger the inferences of new requirements. And these inferred requirements in their turn may lead to the problem of restructuring and the discovery of partial solutions, new goals and new evaluation criteria. They stress and suggest keeping a rationale record of design decisions (design memory) to support designers.

Regarding the iterative process, Jeffries et al. (1981, page 257) see that “each iteration is a representation of the problem at a more detailed level” and this mode of decomposition leads to a top-down breadth-first expansion of design. This is in fact a rather simplistic conclusion. The studies by Malhotra et al. (1980) and Carroll et al. (1979) reveal that the relationship between different design cycles is more complicated. They describe the process as non-hierarchical and involving the discovery of new requirements and the development of partial solutions. Guindon (1990a) criticises their findings as being an expected characteristic directly connected with the presence of the client in the process. Furthermore, the researchers in this early period do not describe
the relationship/connection between designer’s cognitive behaviours (moving between goals/requirements) and the structure of the produced model, especially in the context of the design space and design status.

The systematic evolutionary process of requirements engineering of this early period as described above has been taken up and is described in various contemporary requirements engineering textbooks and courses. The “standard” requirements engineering process is described as a “systematic” and “iterative” process (Loucopoulos and Karakostas, 1995, page 13) of elicitation, specification and validation (Loucopoulos and Champion, 1989; Loucopoulos and Karakostas, 1995; Macaulay, 1996; Christel and Kang, 1992; Simsion, 1994).

**Opportunistic perspective**

Recent research (Guindon, 1989; Guindon, 1990b; Visser, 1988; Visser, 1992; Visser, 1994b; Visser, 1994a; Davies, 1991; Khushalani, 1997; Carroll and Swatman, 1999b) postulates and demonstrates the opportunistic nature of the requirements engineering process. According to these authors, the requirements engineering process is not smoothly evolutionary; solutions to problems/subproblems are insight-driven rather than through a systematic evaluation of alternatives.

According to Guindon (1990b), the ill-structuredness of the requirements problem is an important factor inducing the opportunistic behaviours of the designer. The human working memory is limited, so mental simulations of the overall solution could not be performed, only simulations of partial solutions. With partial solutions it is possible to keep in memory the sub-solutions’ values, calling others at a different (often lower) level of abstraction (Guindon, 1989; Guindon, 1990b; Guindon, 1990a). The author argues that the inferences of new information reduce the incompleteness and ambiguity of the
problem and lead to opportunistic design process. Often upon suddenly discovering new information, the designer tends to immediately develop new partial solutions, test and modify them, rather than continuing to work on their previous planned task, i.e. higher goals. In addition, in developing partial solutions, design heuristics are often used by retrieving the appropriate standard structures mapped from past solutions in the designer’s repertoire. In addition, the new inferred constraints may provide early insights critical in reducing the space of design possibilities and in discovering decomposition. The traversal between different abstraction levels is described as not systematic.

Guindon (1990a) finds that designers often follow a train of thought, develop and drift through a stream of associated partial solutions in violation of balanced development. She explains that these “partial solutions may provide them critical insights on the proper way to decompose the problem and reduce the daunting size of design possibilities” (Guindon, 1990a, page 333). This conflicts with Adelson and Soloway’s (1985) findings described above. Clarifying these contradictory views, Guindon (1990b) criticises previous work by Jeffries et al. (1981) and Adelson and Soloway (1985) as lacking in observing the inferences and elaboration of requirements. Although acknowledging that work by Carroll et al. (1979) and Malhotra et al. (1980) report these characteristics, she explains that they were, again, due to the context of dialogues between the customer and the designer. Guindon (1990a) also argues and emphasises the central role of the inference and immediate handling of requirements in opportunism of design.

Let us review a number of related studies by Visser and his colleague in order to examine further the opportunistic character of requirements engineering. Although these studies did not involve requirements engineering, they reflect the real practice of complex socio-organisational-technical problem understanding and solving, due to
the real-time characteristic, the length of the studies and the design tasks under study. Therefore, the findings from these studies may be applicable to understanding the cognitive process of requirements engineering.

Visser and Falzon (1988) study expert knowledge elicitation in an ill-structured problem (preparation of elements in compound materials) and observe that “the expert does not solve the problem by following a standard, pre-existing method ...”; “the definition of the problem does not precede processing”; and “a given problem does not have a single solution, but a class of acceptable solutions” (Visser and Falzon, 1988, page 1-2). The authors consider these observations to be characteristic of an opportunistic design process.

Further, Visser (1988) conducts a real-time observation of three designers during their real work activity over thirteen weeks and concludes that designers may find it desirable to deviate from their intended/advocated plans. In this extensive field study, the designer developed a plan for the problem decomposition initially. However, during the design process there were deviations from the intended plan, the design attentions shifted and drifted, and activities were organised opportunistically and guided by the evaluation of the actions. Various designers’ opportunistic behaviours are observed and described, for example as abandoning design components before completing them, returning to and modifying previous components, dealing with later components in advance, interrupting the design and simulating to check a component.

Over three weeks’ observation of a mechanical engineer specifying functional requirements for an automatic machine tool, Visser (1992) views action-management as an important factor underlying the opportunistic character of the design process and that as the designer moves to the local abstraction level, the design process is no longer
top-down decomposition. The author explains that drifting and involuntary attention switching occurs as a result of the designer “looking in various directions”, coming upon information that “obviously applies to another design component”, or taking advantage of the information (Visser, 1992, page 96). He argues that causes of these deviations include cognitive-cost, the importance of action, and a combination of both criteria.

Davies (1991) integrates earlier views and proposes a process model for programming as globally top-down with local opportunistic episodes. Observing and analysing the jumps between and within abstraction hierarchy by various programmers, he judges that the programme design process is “neither strictly top-down or globally opportunistic”. He offers an explanation that “opportunistic episodes may occur at any point in the evolution as a result of simple cognitive failure”, i.e. working memory capacity limitations. This explanation is rather inadequate; it presents only a single reason and does not provide a comprehensive analysis of the opportunistic episodes in terms of how they happen and their impact on the requirements model. In fact, there was no clear evidence (in terms of observational data) to support such explanation in the description of the study and recorded diagrams.

Visser (1994a) considers Davies’s (1991) conclusions as unrealistic due the fact that Davies (1991) examines expert designers performing simple programming tasks with the most difficult task requiring only 78 minutes. Visser (1994a) considers the top-down process as a special case of the opportunistic model. Furthermore, he identifies two types of variables behind the opportunism as situational and processing. The situational type includes, for example, a subjective representation of the problem by the problem solver, problem characteristics, problem structure or structuredness and
subject characteristics, expertise and individual differences. The problem processing type include a sophisticated set of action selection criteria, such as cognitive control criteria, cognitive economy and the importance of action proposed. Visser (1994b) confirms that the organisation of design activities is opportunistic and stresses that the designer deviates from pre-existing plans to satisfy action management constraints with the most important being cognitive economy. These conclusions are concurrent with and more comprehensive than previous empirical studies, for example Jeffries et al. (1981) and Guindon (1990b).

Briefly, later research (Guindon, 1989; Guindon, 1990b; Visser, 1992) observes a large number of deviations from the top-down or stepwise refinement process and casts considerable doubt upon the description of the requirements engineering process as balanced and systematic. These authors conclude that the process is best characterised as opportunistic. Observing that designers tend to adopt a solution rapidly rather than elaborate other alternatives in depth, Guindon (1990b, page 299) also points out that “evaluation criteria, if wisely selected, can effectively reduce the daunting complexity of the design process”. Visser (1994a, page 239) views that the current support systems imposing hierarchical design process “will probably handicap designers”. He suggests that systems which offer real support should assist designers in organisational and representational activities, in management of memory limitations and in re-use through supporting analogical reasoning.

Taking a step further, Khushalani (1997, page 59) proposes a formal and comprehensive definition of opportunism. He offers a formal definition of opportunism as follows:

“Opportunism may be defined as a strategy whereby designers during problem solving discover new goals and activities, and/or adapt their in-
tended goals and activities, in response to the state of the problem and the environment in which that problem exists. The discovery and/or adaptation by designers often results in a changed ordering of goals and activities, omission of goals and activities, and/or addition of new goals and activities in the previously intended path”.

Observing two groups of requirements engineers working with an ill-structured problem, he finds evidences of opportunistic behaviours of designers. A design rationale technique was used for the facilitation of explicit expressions of outcomes of the brainstorming activities of the designers. The observational data reveal that there is little or weak evidence that designers explore more than one variation option at any time, explicitly determine and weight the various criteria for assessing the options and consider consequence of possible options; and they may forget to complete their postponed actions. Khushalani (1997) demonstrates and concludes that it is possible to support the opportunistic behaviours of designers through the provision of an option generation and reporting technique and appropriate training. While this conclusion is rather tentative, his investigation technique and the findings demonstrate that the knowledge of the “why” underlying the design process is profitable, and in fact, essential in understanding the behaviours of the designers.

Nonetheless, the generalisation of conclusions reached by these studies to the complex reality of real world professional practice is not convincing. A common weakness of these studies lies in their artificial setting (often involving simple problem cases over a short period of time, except Visser (1988) and Visser (1992). The analysis of verbal thinking-aloud protocols and retrospective interviews is often done against a small number of preplanned existing variables, for example time slots when the designer works
at a level of abstraction and jumps between different levels. Soloway (1986, page 263) considers that the value of insights gained into programming-in-the-small lies in providing “powerful anchors” to research into programming-in-the-large by formulations of specific issues to be examined. On one hand, thinking-aloud protocols may provide a rich source of data for the identification of basic behaviour patterns and the robustness and diversity of behaviours. On the other hand, it is acknowledged that there is danger in being overwhelmed by data, and gathering and analysing data; moreover, the particular choice (especially of simple cases) might not be representative of the population in general (Soloway, 1986). By limiting the observations to a simple case over a short period of time the diversity of behaviours might be incomplete. Moreover, verbal thinking-aloud protocols by designers might not be comprehensive and accurate due to the ad-hoc characteristic. While the designer concentrates on the design activity, he/she may miss important details in expressing their thoughts simultaneously. Therefore, although these studies contribute insights into understanding of the requirements engineering process, the research outcome is not convincing.

Recently, preliminary results have emerged as a series of field studies (which as a companion project to this research) being undertaken with a view to gaining a deeper understanding of, and formalising (modelling) the requirements engineering process (Carroll and Swatman, 1998; Carroll and Swatman, 1999b). These studies strongly suggest that the requirements process does not accord with top-down development; that it is, in fact, more than locally opportunistic. It is found that while the requirements engineer elicits, represents and validates information in an orderly manner, the requirements engineer’s traversal of the problem space is by no means orderly. The requirements engineer starts working in one problem area, gathers some detail, moves
into one or more areas often leaving areas of the problem without even tentatively solving them, eventually returning to the areas previously considered and postponed. The requirements engineering process is described as an unpredictable and adaptive exploration of problem areas, which is "characterised by frequent discovery and/or adaptation of goals and activities, in response to changing circumstances" (Khushalani et al., 1994, page 13). Furthermore, the requirements engineer’s effort in solving the problem is found to be either reflection-in-action through playing out improvisation (Schön, 1983; Schön, 1996), insight—a sudden flash of thought that solves a problem (Mayer, 1992) or the incubation of ideas (Wallas, 1926) (when moving away from the problem) in the hope of reaching a resolution.

In conclusion, there is common agreement throughout the literature that the requirements engineering process is dynamic, evolutionary and involves continuous decisions. In addition, implications from most research suggest that a well-documented design memory and effective management of design issues would be beneficial for the designer. While previous research focuses primarily on the examination of designer’s activities individually in the context of abstraction levels rather than provide a rich contextual analysis of the phenomenon (except for Carroll and Swatman (1998) and Carroll and Swatman (1999b)), they lack the examination of the dynamics conceptual understanding and perception of the problem by the designer. Although Khushalani (1997) profitably uses a design rationale notation to represent and map cognitive concepts graphically, these diagrams do not demonstrate the impact of critical opportunistic decisions on the conceptual understanding of the designer.

Although opportunistic behaviours are observed to be critical in the requirements engineering process, the questions of how they occur and what their triggers are have not...
been described adequately. The idea of insight can also be compared to *breakdown*, a term coined by Heidegger (1967) to refer to the moment when hidden problems become apparent and important questions arise. Although breakdown moments are critical, it is not clear how the critical moments happen. Although there is opinion that breakdowns are very personal and depend on the designer’s past experience (Kaplan, 1990), the designer himself/herself does not know when they are close to the solution prior to these moments (Mayer, 1992). Despite the difficulty in identifying the causes of insights, a contextual and detailed description of the requirements engineering process is possible to capture and is needed.

Finally, the analysis of the process of requirements engineering is still controversial; undoubtedly there is a need for further evidence and consideration. Moreover, the unpredictability of the process (Carroll and Swatman, 1999b) provides a distinct challenge: what can be done to support and improve the requirements engineering process?

### 2.1.5 The volatility of requirements

In being transformed from ambiguous and incomplete statements into a precise and complete specification, requirements evolve and undergo frequent changes. Therefore, requirements volatility is an inherent property of requirements models (Christel and Kang, 1992; Gotel and Finkelstein, 1994). Requirements volatility often causes major difficulties during the development process. Conducting a field study from 17 large software development projects, Curtis et al. (1988) find that fluctuating and conflicting requirements are one of three salient problems in software development. The authors stress that requirements change over time and represent a “*moving target for designers*” (Curtis et al., 1988, page1278). Another field study of ten organisations with 23 projects
conducted by Lubars et al. (1993, page 9) reveals that “changes to requirements can have widespread effects on the rest of the system” and poor management of changes can lead to the failure of systems development projects.

Factors leading to the instability of requirements have been identified. Curtis et al.’s (1988) study exposes a sophisticated set of sources of requirements fluctuation including market impacts, such as customers, technology advances and competitors products; company impacts, such as corporate politics and corporate product lines; and hidden impacts from implementors and managers, such as creeping elegance. Poor communication and the lack of application domain knowledge are also among the sources of fluctuating requirements. According to Lubars et al. (1993), changes to requirements are instigated by customers in the customer-specific projects, or by the market in market-driven projects; by developers when detecting ambiguities, reusing previous experience, or during systems integration. In Christel and Kang’s (1992) view, unforeseen socio-organisational pressures and the acquisition of knowledge during the process result in changes to requirements. Environmental factors, for example government regulations may also cause changes (Macaulay et al., 1990; Curtis et al., 1988; Christel and Kang, 1992).

Briefly, the literature tends to the view that the diversity of wants and needs of different participants, the political climate and the organisational complexity are primary sources leading to requirements volatility. Although changes to requirements due to the nature of the cognitive problem understanding and solving activity have been recognised by some authors, for example, the acquisition of knowledge (Christel and Kang, 1992) and the detection of ambiguities (Lubars et al., 1993), the exploration of this view is still limited. Indeed, the creativeness of the development personnel which is intrinsic to
requirements engineering has received less attention and has not been acknowledged explicitly as a positive constituent of changing requirements. Perhaps, this shortcoming explains why requirements volatility is often considered as an (undesirable) problem. Therefore, requirements volatility should also be studied from perspectives of: changes to requirements themselves, \textit{i.e.} not merely to their representations; the intrinsic cognitive activity of the requirements engineer, and the organisational complexity.

Therefore, requirements traceability is essential for the identification and management of requirements changes. Requirements traceability is the possibility of tracing the life of a requirement throughout the development life cycle. Current empirical research distinguishes different traceability types and directions (Gotel and Finkelstein, 1994; Ramesh and Edwards, 1993; Jarke, 1998). Among them, pre-traceability and post-traceability are most popular. Pre-traceability refers to the ability to follow a requirement from its original form to the specification while post-traceability refers to the ability to trace a requirement from design components to its origin (Ramesh and Edwards, 1993). Therefore, pre-traceability also concerns the traceability of the process itself and this could be made possible by capturing process knowledge (Pohl, 1994).

Research has been undertaken to investigate requirements traceability and its benefits, issues and use in an attempt to build up a comprehensive model for requirements traceability and to solve traceability problems, for example Gotel and Finkelstein (1994), Ramesh and Edwards (1993), Ramesh (1998) and Jarke (1998). This has resulted in various approaches to alleviating the traceability problems. Design rationale\textsuperscript{1}, or the deliberation behind the design process has been utilised to address the traceability problem. Examples of this are IBIS (Issue Based Information Systems (Conklin and

\textsuperscript{1}The use of design rationale in requirements engineering will be discussed in details in section 2.2
Yakemovic, 1991) and COED (Conversation-Oriented Environment for Design), a tool for recording and coordinating collaborative activities (Kaplan, 1990). Another approach to requirements traceability is version control and configuration management, for example Macfarlane and Reilly (1995). Weaknesses of these approaches lie in two areas, either a weak connection between design components and associated explanation or justification (structured using a semi-formal or formal notation), or a lack of support for other requirements engineering activities.

There are other approaches which aim to capture the process knowledge and associating it with the artefacts. For example, REMAP (REpresentation and Maintenance of Process Knowledge) has been designed for maintaining and representing process knowledge (Ramesh and Dhar, 1992). Ramesh and Luqi (1993) describe a support tool for capturing structured history of the requirements process, CAPS. EOM (Evolutionary Object Model) has been presented as a generic model to structure the requirements engineering process (Moreno and Souveyet, 1993; Rolland, 1994). In this model, the requirement model is seen as an Evolutionary Object and design rationale is recorded and stored in the Evolutionary Object’s history. Other authors emphasise process traceability as well as requirements traceability, for example NATURE (Jarke et al., 1993) and PROVE (Rose, 1998). These approaches are criticised as lacking a model that is established prior to the use, consequently leading to an “unwieldy mass of unstructured and unusable data without a prior discrimination concerning the type of requirements information that practitioners are likely to need and for what purposes” (Gotel and Finkelstein, 1997, page 170).

Contribution Structures is an approach proposed and developed with a view to modelling networks of requirements engineering personnel including their social roles and
role relations (Gotel and Finkelstein, 1995; Gotel and Finkelstein, 1997). This approach claims to extend artefact-based requirements traceability with personnel-based requirements traceability. However, a weakness of the approach is the lack of a sophisticated and sufficient support for the requirements engineering process.

Although these approaches and recent traceability tools have made significant contributions to requirements management, there are still major gaps (Jarke, 1998). Reviewing the current state of the art, Domges and Pohl (1998) state that existing traceability tools show common weaknesses in their incorporation within the requirements engineering process, adaptation to the situation, and support for the creation of organisational knowledge.

Analysing and synthesising recent views in requirements engineering and requirements tracing, Jarke (1998) argues that requirements traceability should be examined and supported with different perspectives: as a corporate strategy, as a product and as a process. He proposes a traceability meta-model which captures three dimensions of requirements engineering (Pohl, 1994). With regard to the process perspective, he criticises current traceability methods and tools claiming they “support the necessary flexibility in trace capture and usage only in a rudimentary manner…” (Jarke, 1998).

Despite the growing number of studies, methods and tools, requirements traceability is still poorly addressed and implemented in practice. Lubars et al. (1993) find that many companies tend to rely on the expertise of developers in dealing with changes to requirements and only several companies explicitly perform configuration control over the requirements documents. Ramesh (1998) conducts a series of empirical studies over four years. This research programme is the most comprehensive survey of traceability practice to date. Two groups of low and high-end users of traceability are examined and
factors influencing their practice are identified. Low-end users merely see the maintenance of traceability as obligatory for standard compliance, they use ad-hoc approaches, identify simple schemes and create only static documents. High-end users, however, see traceability as a means to ease the task of life-cycle maintenance, to achieve long-term improvement and to gain competitive advantages. These organisations develop well-defined traceability policies as an integral part of software development (see also Ramesh et al. (1995)). They also use traceability to facilitate understanding and to improve the process. Although the creation and maintenance of dynamic traceability information is appreciated, the difficulty in management of the large volume of information is a clear concern.

Overall, the following issues emerge from the above discussion:

- Current methods and tools assume an incremental evolution of requirements. If the requirements evolution is not incremental then methods and tools may not support requirements engineers in the management of changes as expected. Specifically, the opportunism and the creativity found from cognitive studies have not been integrated with current methods to support the problem understanding and solving activity of designers.

- The iterative process is shown to be essential to address the requirements volatility problem. Requirements are discussed, refined and stabilised through an iterative process of negotiation between the developers and the clients (Curtis et al., 1988; Lubars et al., 1993; Macaulay et al., 1990; Christel and Kang, 1992). As argued by Macaulay et al. (1990, page 102), the iterative process enables the solutions to be “reworked in the light of increased knowledge”.

However, what is clearly missing in the literature is a description of a parallel
evolution process of different views contained in various representation forms of
the requirements model. The literature tends to describe the requirements as
evolving from an opaque state to the final specification, from informal to semi-
formal and formal notations (Swatman and Swatman, 1992; Pohl, 1994; Siddiqi
et al., 1994). Swatman and Swatman (1992) also describe a cyclic process of
informal modelling, formal modelling and debate and validation of requirements.
The primary contribution of this cyclic process lies in highlighting conflicts and
issues of requirements, therefore, and in providing opportunities to solve them in
further cycles. This is further refined as a result from an action research study
by Fowler and Swatman (1997). This study explicitly shows that the require-
ments engineering process is cyclic and each cycle involves informal, semi-formal
and formal modelling subprocesses loosely coordinated and in parallel (see also
section 2.3). Indeed, most requirements engineering methods involve different
views of requirements, expressed in different representation forms and evolving
in parallel over a long period of time. For example, in Booch’s Object-Oriented
approach (Booch, 1994), requirements can be expressed as static structure mod-
els and/or interaction models. This description is in agreement to what Ramesh
and Edwards (1993) call horizontal traceability. Horizontal traceability, as iden-
tified from their empirical study, refers to the correspondence of requirements
transformation between different subprocesses at the same phase of the life cycle,
for example between different representation forms. Vertical traceability, on the
other hand, refers to the correspondence between different design components in
different software life cycle phases. However, no solution has been proposed.
These representation forms reflect different views (perspectives) but they all con-
cern the same underlying system. Consequently, these requirements models may
evolve in parallel while still maintaining certain semantic correspondence with each other. The process of parallel evolution of different views of requirements is rather complex and needs further observation. Furthermore, how to support this process is a challenging and open issue in requirements management.

• Documenting the development process is reported to be beneficial in controlling and improving the traceability of evolving requirements, particularly in facilitating “understanding the evolution of the system as well as identifying the choice points where the alternative decisions could lead to different paths” (Ramesh and Luqi, 1993, page 251). Many approaches and tools have been developed to support the capturing of process knowledge and/or integrating this knowledge with design components. Nevertheless, requirements engineers face the problems of understandability and maintainability of huge stocks of data. Clearly, there is a strong need for an approach which would organise a large base of process information and manage the documentation activity in the context of evolving requirements. Last and not least important, process knowledge may often be created in an ad-hoc manner, hence, the interrogation of the recorded data should also be addressed. “One has to design data reduction and abstraction facilities” (Rose, 1998, page 52) for supporting the maintenance of the huge amount of process knowledge and for effectively making use of it.

2.1.6 Requirements engineering as a social and communicative process

The problem of requirements engineering cannot be solved purely with technical skills; social and communication skills are also needed. Traditional requirements engineering
methods draw upon philosophically functionalist assumptions. Indeed, various modelling approaches and techniques, such as structured systems analysis, data flow diagrams, data-oriented approaches, Object-Oriented approaches, and formal specification languages, focus on producing functionally correct and efficient requirements. More recent research streams offer a variety of systems development approaches with different philosophical assumptions which demonstrate the growing recognition of social and communication aspects among the Information Systems community (Hirschheim and Klein, 1989; Macaulay et al., 1990; Goguen and Linde, 1993; Bickerton and Siddiqi, 1993; Jirotka and Goguen, 1994).

The requirements engineering process is social. There are different groups of participants involved in the process, such as clients, managers, requirements engineers, developers/implementors and end-users. Therefore, the understanding and representation of requirements depend very much on how each party interprets requirements. Checkland and Scholes (1990) argue that requirements must be interpreted and understood in the socio-organisational context within which the system will be used. Given the subjectivity of the interpretation of requirements, the complexity of organisational environment and the diversity of weltanschauung of different communities/stakeholder groups involved there may be different and conflicting viewpoints of requirements. Moreover, there may be misunderstanding between the requirements engineer and the client due to terminology differences. In fact, Dawson and Swatman (1999) find that clients often have problems in understanding diagrams in semi-formal notations, such as object-oriented diagrams, used by the requirements engineer. This makes the validation of requirements models difficult. Carroll and Swatman (1998) suggest a theme of mutual education. Their interpretation of a situated observation of a real project suggests
that the requirements engineers educate the clients about possible business and technical solutions while the clients educate the requirements engineers about their specific business, domain and expectations. Therefore, the requirements engineering process involves communication, negotiation and integration of viewpoints; then bridging the disparities between them (Christel and Kang, 1992). Pohl’s (1994) model reflects this as the agreement dimension of requirements engineering.

Various approaches and techniques using sociological concepts and methods have been developed to address the social and communicative issues in requirements engineering. The following discussion illustrates these approaches.

Grounded in the fundamental concepts of hermeneutics and weltanschauung (more details will be provided in Chapter 3), Checkland and Scholes (1990) develop the systems approach Soft Systems Methodology (SSM) which elicits the complexities and represents the emergent properties of a real world problem situation. The main steps of the approach include: understanding the problem situation; building a rich picture of the real world; producing root definitions of the real world by selecting a viewpoint and using mnemonic CATWOE (Customer, Actor, Transformation, Worldview, Ownership, Environment); building and matching conceptual models for root definitions with the real world; implementing desirable changes; and identifying actions for improving the situation. The approach, therefore, is effective in constructing a socio-organisation contextual analysis. Taking a step further, many authors attempt to integrate SSM into detailed systems design or formalise root definitions. For example, Lewis (1993) argues that an interpretative form of data analysis can be used within SSM and that a close integration of SSM with data-focused approaches is theoretically feasible and may be practically desirable. Later, Houlihan et al. (1996) critically describes and examines
the Logico-Linguistic approach to developing a conceptual model from SSM root definitions using formal logic. More recently, Lamp (1998) explores and demonstrates an application of Petri Nets in order to formally describe and develop a conceptual model which takes into account different *weltanschauungen* (worldviews) expressed in SSM without losing their richness. Nevertheless, the difficulty in this integration is often acknowledged as a weakness of the approach.

The socio-technical approach tries to unify user needs, user participation, organisational issues such as organisational structure and job satisfaction, (Jirotka and Goguen, 1994) throughout the life of a development project. Mumford (1985) describes the participative methodology Effective Technical and Human Implementation of Computer-based Systems (ETHICS) which claims to be a simple step-by-step guide to defining requirements. Requirements are defined from an analysis of the organisation tasks and problems. This analysis then forms the basis for setting the business objectives. Finally, it is translated to the systems requirements. The approach involves, and very much depends on, user participation and commitment to the system. The authors argue that user involvement produces good data and ensures the business interests. Inspired by *neohumanist* values, Hirschheim and Klein (1994) judge that emancipation could be incorporated and achieved within ETHICS (Mumford, 1985). Nevertheless, the approach is criticised as neglecting “*how people actually carry out the activities that they do in the work place*” (Jirotka and Goguen, 1994, page 6). Multiview (Avison and Wood-Harper, 1986: Avison and Wood-Harper, 1991) and ORDIT (Organisational Requirements Definition for Information Technology) (Dobson et al., 1994) are other examples of this approach.

Ethnography can play an important part in requirements capture and analysis. Som-
merville et al. (1993) describes a project in which an ethnographer spent several months in an organisational environment observing the practices, communication and processes. The authors, however, also point out that the contribution by the ethnographer is still limited. This is due to the distance between philosophical assumptions of ethnography and structured analysis methods and also the extreme difficulty in incorporating the two disciplines. The authors suggest that support tools are needed and effort and flexibility from both software engineers and sociologists are required. Further useful discussion of the use of ethnography in requirements engineering is offered by Randall et al. (1994), who provide an evaluation of the contribution of ethnography to the development of air traffic controllers.

There are a number of other approaches to representing and integrating different participant viewpoints. Based on an earlier, user-centred approach, USTM (Macaulay et al., 1990), Cooperative Requirements Capture has been developed with a focus on reaching shared understanding and cooperation between stakeholders. In contrast, Easterbrook (1994) argues for the need for explicitly detecting, negotiating and managing conflicts within different viewpoints. The author then suggests a support tool (Synoptic) to address these problems utilising a design explanation notation. Darke and Shanks (1996) also developed a model for the organisation and representation of user viewpoints. Flynn and Jazi (1998) developed the approach of user-lead requirements construction (ULRC) whereby the clients are trained and build requirements models themselves. While emphasising user-oriented and communicative factors, these approaches lack formality and support for problem understanding and solving activities. An environment which integrates this support with the social approaches may be profitable.
Other authors develop different approaches to facilitating collaborative work and communication using design explanation (Kaplan et al., 1992; Hawryszkiewycs, 1993; Johnson, 1996). For example, design explanation has been incorporated successfully within a formal specification method to represent the rationale behind formal documents which are often hard for the client to understand (Johnson, 1996). In his study, the use of design explanation is limited to explaining requirements models expressed using a formal notation. However, its use in understanding organisational context and supporting mutual education and negotiation between the specifier and the client and within the team of specifiers during requirements modelling have not been adequately explored.

FOOM (Swatman, 1996) is an approach which focuses on the incorporation of socio-organisational contextual analysis and structured (semi-formal and/or formal) approaches within requirements engineering. FOOM will be introduced and explained in detail in section 2.3.

Overall, research takes different directions in integrating the social and human aspects within requirements engineering (Jirotka and Goguen, 1994). The approaches of the first direction tend to introduce additional (social, organisational or contextual) analyses and later to integrate them into existing requirements methods. The approaches of the second direction encourage the clients to be involved directly and actively in the requirements process, especially in the early phase of requirements elicitation. In the approaches of the last direction, “the social and the technical are thoroughly intertwined” (Jirotka and Goguen, 1994, page 7) (e.g. using ethnography). Most of the approaches, however, still exhibit their weaknesses in the gulfs between different disciplines and their assumptions. Specifically, with the exception of FOOM (Fowler and Swatman, 1997), they imply a sequential transformation of the client informal towards
more formal forms of requirements and they do not provide sufficient guidance for the transformation of informal analyses into more formal designs, for example when to introduce diagrams and/or formal notations, and how validation can be performed. This thesis does not intend to develop a completely new approach to solving the social and human problems in requirements engineering. However, these factors will be taken into account when investigating the requirements engineering process. In this context, the thesis will focus on the need to have a mechanism for supporting this activity of knowledge sharing and communication between clients and specifiers and within a team of specifiers.

2.1.7 Discussion

The management of the requirements engineering process is inevitably an essential part of project management and plays a critical role in the success of a project. Theory suggests that quality and process management can be achieved through deep understanding and continuous improvement (Deming, 1986; Humphrey, 1990). Studying quality in various fields, Deming (1986) argues that if the process is stabilised (or is in statistical control) then repeating the process will produce approximately the expected results and improvement of the process can be effective. Humphrey (1990) suggests a model for software process management consisting of five levels of process maturity. He advises that the main objective is to achieve a controlled and measured process as the foundation for continuing improvement. Although measurement is the basic principle of statistical control, the numbers measured must be meaningful (Humphrey, 1990). This suggests the need for understanding and managing the requirements engineering process. However, there are a number of issues as follows.
As discussed above, requirements engineering is a crucial phase in a systems development life cycle. Requirement engineering is a problem understanding and solving activity where the problem is ill-defined. Thus, the process is a complex cognitive activity. The problem exploration and the solution path are described as creative, opportunistic and unpredictable, but it is still not clear how the requirements engineering personnel use their knowledge and creativity. In addition, the process is dynamic and involves the evolution of requirements embedded in different viewpoints of different participants. As argued in section 2.1.5, the product and process traceability needs further investigation and support. The management (control and measurement) of such a process, consequently, presents a very difficult and challenging task, especially, since the literature shows that the requirements engineering process is still poorly understood.

There are questions about current software development methods systems development life cycle models. Only a few “include process components identified in empirical research on design problem-solving (Curtis et al., 1988, page 1269). The most popular models are the waterfall model (Boehm, 1976), the spiral model (Boehm, 1986; Boehm, 1988) and the fountain model (Henderson-Sellers and Edwards, 1990).

The waterfall model has been adopted as a general standard systems development life cycle by many software developers. This model directly supports functional decomposition. Initially, the waterfall model describes the software development process as a linear series of different phases (for example analysis, design, implementation and maintenance) in which one phase must be completed before the next phase can take place. In an extended version, feedback between two consecutive phases is allowable. Nevertheless, the model is often criticised by many authors as being linear, thus inflexible with a rigidly indicated order of development phases (Henderson-Sellers and
Edwards, 1994; Curtis et al., 1992; Goguen, 1994; Wieringa, 1996). Henderson-Sellers and Edwards (1990) argue that a major problem with the waterfall model lies in the flaws of top-down function decomposition perspective. Consequently, it tends to freeze requirements and does not support evolutionary changes (Henderson-Sellers and Edwards, 1990; Bersoff and Davis, 1991) while volatility are an inherent characteristic of requirements (see section 2.1.5). Another problem of the model lies in the expensive handling of changes to requirements due to the ripple effect on the subsequent phases (Henderson-Sellers and Edwards, 1990; Henderson-Sellers and Edwards, 1994). Moreover, different underlying models used at different development phases (e.g. data flow diagrams, hierarchy charts and flow charts) may lead to disjoint mappings between them. Clearly, the waterfall model does not support “the seamless transition” from one development phase to another (Henderson-Sellers and Edwards, 1994). Therefore, the waterfall model (and any other models which are limited to the view of the development lifecycle as linear) is rather simplistic and less effective.

The spiral model improves the waterfall model significantly by allowing an iterative development process (Boehm, 1986; Boehm, 1988). This model aims at identifying and reducing risks through iterations or feedback loops within the development process. Each spiral of the model may be considered as a waterfall model. In terms of risk reduction, each spiral consists of four activities of objective setting, risk assessment and reduction, development and validation, and planning. Therefore, the model allows iterations and flexibility required for in project management. The model is adopted by various authors, especially in object-oriented software development, (for examples, Wirfs-Brock et al., 1990; Booch, 1994; Sommerville, 1996). However, as acknowledged by Boehm (1988), a pitfall of the model lies in the poor identification of risks. Fur-
thermore, the model may be considered, in fact, as another form of the linear model
(Henderson-Sellers and Edwards, 1994) because it does not allow backtracking and
iterative cycle across previous phases.

Another popular alternative life cycle is the fountain model (Henderson-Sellers and
Edwards, 1990). In the fountain model, the development activities flow upwards from
the requirements analysis phase to programme use through various sequential phases
and may fall down for maintenance. Similarly to the spiral model, the fountain model
also promotes iterations. In addition, the fountain model offers more flexible iterations
and overlaps across more than one phase (two or three). Therefore, a major advantage
of this model lies in allowing both the sequential flow of development phases as well as
the flexibility of iterative cycles across previous phases. Another major advantage the
model offers is fountain lifecycles for each subsystem, thus, it significantly reduces the
ripple effect on other subsystems caused by changes to requirements in one subsystem.
Therefore, the model supports knowledge acquisition and changes management better
than the previous models. However, the model needs to be re-examined as to whether
it promotes the architectural flexibility, i.e. the flexibility of radically restructuring the
system, not just each subsystem/component.

These models, especially the first two models tend to impose an incrementally evolu-
tionary process of requirements development and therefore would not aid the creative
and opportunistic model of problem exploration and problem solving activities. Since
the requirements process is still poorly understood and needs further consideration,
lifecycle, methods and CASE tools imposing these models (being based on current un-
derstanding of the process) may not assist the systems developer as expected. This
strongly indicates that a better understanding of the process of requirements engineer-

49
ing is desirable.

Throughout the literature, it is suggested that knowledge of how and why requirements decisions are made, represented, modified and communicated, and their impact on the further development process is desirable. The benefits are twofold. First, this information is beneficial in understanding the requirements process and improving the understandability and traceability of the requirement specification. Second, in most empirical previous studies, this kind of information has been captured informally and in an ad-hoc way and has been profitably used in understanding and analysing the designer’s behaviours. However, the information, if documented systematically and structured more formally, it would provide researchers with potential benefits in understanding and investigating approaches to controlling and improving the process. Approaches to capturing and documenting this information (design rationale) will be discussed in the next section.

2.2 Design Explanation

2.2.1 What is design explanation?

Design rationale—The essence and history

As discussed earlier, the requirements engineering process is a design process. The design process is often described as the route that the designer travels in producing the design product. It is a cognitive deliberation activity involving the generation and evaluation of solutions. While the design product describes what has been built, design rationale concerns the reasoning behind the design process and explains why the
product has been built with a certain architectural configuration and with a certain
behavioural description. The following discussion is devoted to the background and
conceptual development of Design Rationale.

The notion can be traced back to the philosopher Aristotle’s argumentation model.
The essence of Aristotelian logic lies in the deductive categorical syllogisms with two
concepts of **figure** (including two premises and a conclusion) and **mood** (including uni-
versal positive, particular positive, universal negative or particular negative) and their
combinations. Similarly to Aristotelian logic, modern standard logic is also based on
deductive rules or laws. In addition, it is based on the sophisticated propositional
and predicate calculus, also known as Boolean algebra. This involves propositions and
predicates, the only two allowable values of true or false and a set of operators and
well-defined deduction laws. Hypothetico-deductivism, a rationalistic approach to ar-
gumentation, is a widely adopted method in various scientific disciplines. It is based
on observation of phenomenon and hypothesis proposal and testing. A philosophical
discussion about positivism based on the assumption of deductive logic will be pro-
vided in Chapter 3. While these traditional logical approaches exhibit advantages in
their objectivity, they are often criticised by contemporary authors as being limited
to well-defined deductive situations and as being less effective in inductive and social
situations (Toulmin, 1958; Toulmin, 1972) (also see Chapter 3 for more details).

Criticising the deductive approaches as candid and insufficiently elaborate, philoso-
pher Toulmin (1958) develops a more sophisticated structure of argument by analogy
with jurisprudence. His “pattern of argument”—the Toulmin model, represented in
the graphical form, marks the first step in the field of “modern” design rationale².

²Toulmin’s model will be described and discussed later.
Toulmin (1972) distinguishes three aspects of scientific explanation of scientific concepts: language, presentation techniques and the application procedures of science. He asserts that the second aspect should aim at demonstrating scientific concepts and their relations rather than proving them deductively. In addition, the Toulmin model’s graphical format for representing the structure of logical arguments strongly influences subsequent design rationale notations.

Recently, design rationale research has been drawn from multiple fundamental sources: artificial intelligence, cognitive study and design research. Most ‘root’ concepts in design rationale have emerged from and been formulated in work by Simon (1969), Engelbart (1963), Rittel and Webber (1973), Rittel (1972) and Schön (1983).

In his analysis, Simon (1969), one of the foundation figures of artificial intelligence, compares the design process to searching the design space in artificial intelligence. The author describes the design process as a search tree with each branch being a partial path. He argues that the logic of design is the exploration and evaluation of alternatives (tentative paths) and choosing the most promising. He distinguishes two types of description in understanding and solving complex problems: state description and process description. Further, he argues that problem solving requires “continual translation between the state and process descriptions of the same complex reality” (Simon, 1969, page 112). He concludes that the problem solving process is, in fact, the discovering of the (process) description of the path that leads to a desired goal (final state). Naur (1985) also recognises the importance of clearly documenting the process knowledge behind the design of computer programmes. Later, in the light of Simon’s (1969) and Naur’s (1985) analyses, Conklin and Yakemovic (1991) suggested that design rationale could be considered as an explanation of the path around these
branch points (design decisions) that link the initial state to the final state of the design.

Engelbart (1963) studies factors that limit an individual’s information-handling capability and develops techniques, procedures and systems to enhance the capacity of an individual to comprehend a complex problem and derive a solution to the problem. Engelbart (1963, republished in Engelbart (1987), page 51) describes purposeful structuring as “the important principle in building sophisticated capabilities from basic capabilities”. Concept structuring is identified as an important type of purposeful structuring. The author states that a concept structure is something that can be developed on paper and worked with through conscious thought processes and can be used by problem solvers to communicate to each other. A concept structure “when mapped into a human’s mental structure will significantly improve his capability to comprehend and to find solutions within his complex-problem situations” (Engelbart, 1963, republished in Engelbart (1987), page 54). Presenting his conceptual framework for the augmentation of man’s intellect, the author calls for designing computer-supported notational structures to assist people to overcome their cognitive limitations. In response to his call, the representation of the design process became a focus of research in various research centres in the early 1980s (Shum and Hammond, 1994; Shum, 1996b).

Rittel and Webber (1973) term problems in policy planning as “wicked” because: they cannot be definitely described, they have no stopping rule and no ultimate criteria for the evaluation of solutions, they are unique and involve pluralistic perspectives of the participants. Recognising this, various authors engage in gaining further understanding characteristics and properties of ill-design problems, and in searching for ways to tackle these problems (for examples, see Simon, 1973; Guindon, 1989). Rittel (1972) contends that there may not be scientific bases for confronting such problems and argues in sup-
port of the need for a *second generation* of design methods—an argumentative approach to solving wicked problems. Rittel and Webber (1973) argue that it is argumentation that constitutes the design process in which consensus emerges through explicitly laying out, discussing, debating and negotiating the problem, its alternative solutions and their pros and cons. Kunz and Rittel (1970) developed IBIS to encourage designers to articulate and represent their arguments and decisions. IBIS was pioneering and “*strongly influenced the adoption of semiformal argumentation as a representation basis*” for design rationale (Shum et al., 1997, page 99). It is worth noting that later, hypertext technology has emerged as the dominant medium for the representation of design rationale (for examples, see Shum, 1991a; Conklin, 1987; Conklin and Yakemovic, 1991; McCall et al., 1990; Pries-Heje, 1993).

From a social and ethnographical point of view, Schön (1983) adds the reflective characteristic of the design activity. The author describes the design process as a continuous alternation of the mutually exclusive modes of *knowing-in-action*, *reflection-in-action* and *reflection-on-action*. I shall not discuss this work in more detail here, except to note that the author emphasises the reflective nature of the design process, whereby the designers reflect on the construction of the problem and their actions in order to improvise in the face of uncertain situations, and “*evolve their way of doing it*” (Schön, 1996, page 173). Schön’s (1983) theory of design inspires and encourages other authors in understanding design rationale and supporting reflection by the designer in his/her work (Fischer et al., 1991; Shum, 1996a; Moran and Carroll, 1996).

At present, design rationale has been developed and used in a wide spectrum of domains, such as legal decision making (Marshall, 1989); linguistics (Botha, 1970); planing and information policy formulation (Kunz and Rittel, 1970); and general design (for exam-
ple, see McCall et al., 1990; Fischer et al., 1991; Chung and Goodwin, 1994). What is important to this thesis is that the interest in design rationale has also been gradually growing in the areas of software engineering, especially in human-computer interaction, (for example Conklin and Begeman, 1988; Lee, 1990; Kaplan, 1990; Lee and Lai, 1991; Conklin and Yakemovic, 1991; MacLean et al., 1991b; Carroll and Rosson, 1996; Pries-Heje, 1993; Johnson, 1996; Moran and Carroll, 1996; Shum et al., 1997). In requirements analysis and modelling (excluding user-interface specification), however, the use of design rationale is still at an early stage (Moran and Carroll, 1996; Greenspan, 1993; Potts and Bruns, 1988; Ramesh and Dhar, 1992; Jarke et al., 1993; Potts et al., 1994; Rolland, 1994; Easterbrook, 1994).

From design rationale to design explanation

Various authors have offer different definitions of design rationale. Many authors consider design rationale to be a record of the design history, therefore design rationale can be seen as a part of design (Potts and Bruns, 1988; Potts, 1989; Conklin and Yakemovic, 1991; Kaplan, 1990; Vanwelkenhuysen, 1995; Chung and Goodwin, 1994). Other authors view design rationale as “a synonym for argumentation” (Fischer et al., 1991, page 395) explaining and justifying the reasoning, therefore, it is a co-product of design (MacLean et al., 1991b; Fischer et al., 1991; Shum, 1996b; Shum et al., 1997). Lee and Lai (1991, page 257) see design rationale as “an explanation of why an artifact is designed the way it is”. Shum (1996b) terms argumentation-based design rationale to refer to approaches to expressing the reasoning behind the design process as arguments about issues. In Dix et al.’s (1993, page 180) analysis, design rationale is considered as “an activity of both reflection (doing design rationale) and documentation (creating a
Recently, Moran and Carroll (1996) synthesised existing views and offer a formal definition of design rationale. This is the most comprehensive definition in the literature and covers six different aspects of design rationale. These include:

- actual reasons for the design of an artefact by the designer
- a design justification produced with the purpose of convincing the client;
- a representation notation for the logical reasons
- a method for designing where design rationale provides the designer with a process-facilitation tool and supports the deliberation “tactically, when it will benefit the process” (Moran and Carroll, 1996, page 9)
- different aspects of design documentation ranging from reasons, design stages, and history of the design process and its organisational, social, political and cultural context
- the explanation of why the artefact is the way it is

In the context of understanding, monitoring and improving the requirements engineering process, the use of design rationale in the first, fourth, fifth and last aspects highlighted by Moran and Carroll (1996) are primary and have a higher priority than the other aspects.

Design rationale, in this thesis, is defined as information which represents and explains the reasoning behind the requirements engineering process. Therefore I prefer to use the term design explanation to refer to design rationale in this thesis.
2.2.2 Approaches to design explanation

For the last two decades, design explanation has received growing attention in the research community. Various approaches, techniques and notations for capturing and representing design explanation have been proposed and developed. They are classified as three strategies (Dix et al., 1993). The two most popular strategies are *ad hoc* (process-oriented) and *post-hoc* (structured-oriented). In the former, design decisions are structured around specific problems and are recorded in the chronological order in which they are made. In contrast to this, the latter concentrates on the logical design space around an artefact, which can be restructured by the *post hoc* consideration of its alternatives. In other words, while approaches of the two strategies may share some similarity in their notations, their processes and foci are different. In the *ad hoc* approaches, as time progresses and design decisions are made, each design discussion is non-intrusively recorded and coded using a notation, such as IBIS (Kunz and Rittel, 1970; Conklin and Yakemovic, 1991). Each *ad hoc* argumentation document explains a decision within the context of a specific problem at hand. However, in the *post hoc* approaches, argumentation documents can be constructed only at intervals, typically between phases of the active designing process. Each *post hoc* argumentation document represents a retrospective and logical examination of the design space around the artefact (MacLean et al., 1991b). These two strategies, their approaches, notations and empirical evaluation will be elaborated in following subsections.

The third strategy is psychology-oriented and is introduced by Carroll and Rosson (1991). The authors pursue psychological interests such as those introduced by Simon (1969) who views the design artefact as an interface between the substance of the artefact itself—the “inner” environment and its surrounding “outer” environment.
He argues that both “computer and [human] brain... are adaptive systems, seeking to mold themselves to the shape of the task environment” (Simon, 1969, page 54). In other words, their behaviour over time is largely a reflection of the complexity of the environment. Based on this assumption, Carroll and Rosson (1991) propose an augmented task-artefact framework for design in human-computer interaction. According to the authors, the tasks identified from observation of how people use an artefact can generate new requirements guiding the evolution of the artefact. Further, they develop their psychological approach to design explanation. Their approach attempts to capture the psychological claims of usability in the artefact in order to support further design work and enhance design to suit the users’ tasks. In contrast to the other design explanation strategies described above, this strategy does not capture intentions and deliberation of the designer during the requirements engineering process. Therefore, this approach is not relevant to my research objectives and further discussion of the approach is out of the scope of the thesis.

The following subsections will critically review different approaches to design explanation and their strengths and weaknesses, reported in previous empirical studies. Although many approaches are originated and developed in the different domains other than requirements engineering or systems development, their implications are valuable and may be considered for research in requirements engineering. Among them, IBIS and QOC are described and reviewed in more detail because they and their central ideas of capturing design explanation are essential to the research project reported in this thesis. This critical review will justify the investigation of the use of design explanation within requirements engineering and the choice of specific design explanation approaches used in the investigation.
2.2.3 Issue-Based Information Systems (IBIS) and its descendants

IBIS

As discussed above, (Rittel, 1972) calls for a second generation of design methods to tackle the wicked problem. The author argues that “the design process is not considered to be a sequence of activities that are pretty well defined and that are carried through one after another and that approaches taken by “the typical design model of the first generation” are not effective in dealing with wicked problems (Rittel, 1972, republished in Rittel (1984), page 321). The author contends that difficulties of dealing with a wicked problem throughout the design process depend on “the state of the understanding of the problem” (Rittel, 1972, republished in Rittel (1984), page 321). Further, he characterises the design process as argumentative; at the micro-level it consists of the generation of solutions to a specific design issue and the negotiation of their pros and cons. This micro process in turn will trigger further issues to be discussed and debated.

Further Kunz and Rittel (1970) propose and develop IBIS (Issue-Based Information System) in response to the challenge of the wicked problem. It is as a method for planning and representing design meetings. The IBIS notation is used to capture and record design argumentative discussions as they occur. In this sense, IBIS supports the ad hoc approach to design explanation.

Figure 2.1 depicts the IBIS model. The IBIS notation takes the form of design questions or problems (Issue\(^3\)), possible answers (Position) and arguments (Argument) for and against the positions (see Figure 2.1). The central activity is the generation and evaluation of positions using arguments. The decision to select a position or discard

\(^3\)Words in Sans Serif such as Issue, Position and Argument refer to elements of a design rationale notation, not in general meaning. See notational conventions, Page xvi.
all but one position resolves an issue. An issue may question, generalise or specialise another issue, and that may lead to other sub-issues with positions and arguments. In Rittel’s IBIS the inter-issue relationship may include, for example, “more general than”, “similar to”, “replaces”, “temporal successor of”, “logical successor of”.

IBIS offers a simple argumentation notation. It is driven by an intuitive guidance: a problem being solved, a set of possible decisions, a number of arguments supporting or objecting to the alternatives, and a commitment to a resolution by not simply counting number of supporting arguments but by a sophisticated evaluation. In other words, it provides the problem solver with a rhetorical model within which ”legal” moves are made in the IBIS design conversation (Yakemovic and Conklin, 1990). It supports monologues of the designer arguing with himself/herself as well as dialogues between the designers. Therefore, IBIS has received growing attention of the research communities from various disciplines and has been taken up in a wide range of applications and domains.

Indeed, it was first developed for government administration and civic planning (Kunz and Rittel, 1970), since then it has not only been successfully tested and widely used in non-design domains, such as in city planning and policy making at the World Health
Organisation (Conklin and Begeman, 1988), by a German inter-departmental government committee dealing with a national plan for information networks, at the United Nations, at the Commission of European Communities, and by the German Parliament (Kunz and Rittel, 1970; Fischer et al., 1991), but it has also been taken up and further extended in design domains, such as in chemical plant design (Chung and Goodwin, 1994), in architectural design (McCall et al., 1990; Fischer et al., 1991) and in software development (for example Conklin and Begeman, 1988; Conklin and Yakemovic, 1991; Pries-Heje, 1993).

Numerous IBIS descendants have extended IBIS to make it suitable in diverse situations and to address different aspects of design needs, such as gIBIS (Conklin and Begeman, 1988), itIBIS (Conklin and Yakemovic, 1991), rIBIS (Rein and Ellis, 1991), PHI (procedural hierarchical IBIS) (McCall, 1991) and REMAP (Ramesh and Dhar, 1992). They are described and discussed below.

2.2.4 gIBIS and itIBIS

gIBIS (Conklin and Begeman, 1988) and itIBIS (Yakemovic and Conklin, 1990; Conklin and Yakemovic, 1991) were developed to extend IBIS for the purpose of capturing design history in systems development. gIBIS represents IBIS design explanation more visually in the form of graph while itIBIS represents IBIS design explanation simply in a textual form. From the notation perspective, the extension of gIBIS (graphical IBIS) to IBIS includes additional elements: Other for a flexible categorisation of argumentation components, External for design documents and codes which are non-IBIS and more flexible involute relationships for Position and Argument (Figure 2.2). A major contribution of gIBIS lies in its graphical hypertext interface with the extended IBIS
semantic types. gIBIS has been developed to support the building and navigating of a large IBIS information base and to assist the computer-mediated collaboration of designers in which design discussions are distributed via computer networks (through a bulletin-board system). The indentation in itIBIS represents the hierarchy of Issues denoted as I. *I represents an Issue which has been solved. ?P represents a Position being considered. *P represents an accepted Position. -P represents a rejected Position. AS and AO represent supporting and objecting Arguments respectively (see the example below). itIBIS may be used irrespective of the technology available—not even a word-processor is required.

![gIBIS Model Diagram](image)

**Figure 2.2: gIBIS model (Conklin and Begeman, 1988)**

The following is an itIBIS example:

*I* : Which processor should be used?

?P : Processor A

\[ AS : Fast \]

?P : Processor B

\[ AS : Already in use, thus cheaper. \]

?P : Processor C

\[ AO : Won’t be available in time. \]

An itIBIS example (Yakemovic and Conklin, 1990)
gIBIS and itIBIS were used by researchers at MCC (Microelectronics and Computer Technology Corporation) and other companies. gIBIS and itIBIS were implemented on a number of engineering computer environments and were used successfully in an engineering project, NCR—hardware and software development for a special purpose workstation controller. In this project, itIBIS was used to collect and structure document analysis, design meetings, and personal brainstorming. The itIBIS notation was found by users to be simple and useful for capturing design notes with minimal changes to their existing work practices. In fact, the introducing and training with itIBIS was quick and simple. A major problem of itIBIS was an unmanageable amount of data scattered across many files leading to the difficulty in finding and updating information. In an attempt to address this management problem, itIBIS files were converted to the gIBIS format where design explanation arguments could be arbitrary grouped and cross-linked. The conversion was found to be time-consuming and therefore, only a number of itIBIS documents were selected, reviewed and converted to gIBIS. This project successfully demonstrates the feasibility and usefulness of design explanation in systems development.

In general, the gIBIS and itIBIS approaches are evaluated as a productive vehicle for research into process-oriented design rationale (Conklin and Yakemovic, 1991). These issue-based approaches are non-intrusive in capturing the design process because they are simple to learn and use. Design explanation information is useful to both developers as well as to maintainers. These approaches (mostly itIBIS (Conklin and Yakemovic, 1991; Yakemovic and Conklin, 1990) show their usefulness primarily in group decision support, conversation structuring and management of group memory in terms of knowledge sharing and refreshment of previous decisions. A major benefit of IBIS lies
in the detection of design flaws from the inspection of the IBIS base. The approaches, as argued by the authors, could be extended to assist in project management and to coordinate action across project groups.

Assessing the usefulness of these IBIS descendants, particularly itIBIS, in the requirements engineering phase, the authors find that design explanation assists the systems analyst in problem understanding and in proposing and evaluating possible solutions based on supporting and objecting arguments. itIBIS results in more productive meetings due to the provision of a set of issues, and perhaps, the refreshment of previously discussed positions and arguments which support communication between the developers. Recorded information by itIBIS and/or gIBIS allows the design to be reviewed from different aspects and therefore improves the design product. The maintenance of design explanation pays for itself. Yakemovic and Conklin (1990, page 113) argue that “if design rationale is documented in an IBIS(itIBIS), the process of performing the review and update this information may pay for itself, by allowing more problems to be found earlier in the development cycle, when they are less costly to repair”.

With regard to the reorganisation of the large itIBIS information base, Conklin and Yakemovic (1991) argue that the primary benefit of the reorganisation is the detection of design errors identified during the conversion of design explanation from the linear form to the graphical form. The savings due to the detection of errors was assessed as being “between three and six times greater than the cost of using gIBIS” (Conklin and Yakemovic, 1991, page 375). However, this conclusion may be rather limited and may needs further investigation on the impact of the reorganisation of the process knowledge on the designer’s problem understanding and consequently, on the design products and process. In addition, the issue whether the conversion solved the management problem
was not reported. This is important because a previous study reports that weaknesses in

gIBIS include inadequate interface and representation (when there are too many nodes
to be represented) and weak links to design artefact (Conklin and Begeman, 1988).

Conklin and Yakemovic (1991) also argue that the capturing of design explanation
also supports opportunistic design process as it allows the designer to move between
different abstract levels or to leave unsolved design issues in order to work on other
issues. This proposition is a very interesting view, however, rather than being explored
adequately, it is considered as one of the arguments in the justification of the cost of
technology transfer.

Reviewing existing requirements elicitation methods, (Christel and Kang, 1992) con-
sider gIBIS and itIBIS as potential methods for eliciting and exploring requirements
in an early phase of requirements engineering. In their view, IBIS is preferable for its
simplicity, non-intrusiveness and support for focused thinking and communication of
the requirements engineer. However, the authors suggest that IBIS issues should be
linked to the evolving design to promote traceability. They also criticise IBIS as not
effective in organising information. Indeed, the capturing of the deliberation as it hap-
pens may result in a poorly organised design explanation document which is difficult to
be accessed and reused. Emphasising the descriptiveness of gIBIS and itIBIS, Conklin
and Yakemovic (1991) acknowledges IBIS is not intended to be reused. They clearly
state that in the process-oriented approach, the design rationale “is merely descriptive;
its reusability is incidental” (Conklin and Yakemovic, 1991, page 368).
rIBIS

rIBIS (Rein and Ellis, 1991) is claimed to be a descendant of gIBIS. It is a real-time hypertext system which is based on the IBIS notation. It is designed to support group design activity through group brainstorming and group decisions making by allowing designers to record and edit the issue hypertext network in real-time. However, the tool was found difficult to use due to lack of experience of the users in the IBIS notation and the complexity of the rIBIS interface. The primary conclusion of the authors is that the more people use the tool, the higher user’s acceptance becomes. Although rIBIS can be considered simply as an implementation rather than an extension to IBIS in terms of notation and/or concept, an important lesson learnt from this study is that the simplicity of the IBIS is a critical usability factor. In this thesis, investigating the IBIS concept is preferable to simply extending notation or developing complicated interface of a support tool.

Linking IBIS to requirements

More recently, IBIS is also implemented in Pries-Heje’s (1993) study for managing problem structuring and design discussions in software development. At first a HyperCard prototype is developed and is evaluated in an empirical study where it evolves through five thinking aloud sessions. The study identifies a number of problems and issues in using IBIS. They are:

- the flexibility of the notation is needed: a solution can solve a number of issues rather than just one issue as in the original IBIS notation;
- scenarios are not included in the notation but they are important; in addition,
there is a hierarchy of scenarios;

- the maintenance of the HyperCard system is time-consuming and tedious.

With regards to the first two problems, the author extends IBIS where Position can be grouped into a Solution package and each Issue is linked with a Requirement which is connected to Scenarios and subsequently to the client’s Want (see Figure 2.3). Although the study proposes a rather simplistic model to solve the problems, it strengthens the claim that design explanation may provide an improvement to the current practice of software developments. However, the author points out that further investigation is required.

![Figure 2.3: An extension to IBIS (Pries-Heje, 1993)](image)

With regard to the maintenance of their system, a prototype is later implemented using the textual form of itIBIS instead of using HyperCard. This approach is different from Conklin and Yakemovic’s (1991) approach (described above) to solving the maintaining problem which attempts to convert itIBIS information into a graphical form (gIBIS). These approaches tend to view and solve the problem simply from the representation
perspective and therefore, lack formal constraints between design problems, assumptions and solutions. A more sophisticated approach to presenting IBIS information and connecting it to requirements and the design is REMAP (Ramesh and Dhar, 1992). REMAP is described below.

**REMAP**

Criticising IBIS as being limited to the local context of argumentation and as being implicitly linked to the artefact, Ramesh and Dhar (1992) go on to develop REMAP (REpresentation and Maintenance of Process Knowledge), an approach to capturing and representing the process knowledge and associating it with design components (Design Object). REMAP is based on an extended IBIS notation. It focuses on task-specific analysis and facilitates the reasoning mechanism.

The REMAP model is more complicated than the IBIS primitive model (see Figure 2.4). The additional components include:

- **Input for the task:** Requirements or a subset of the client’s requirements. They generate issues or sub-goals, which are discussed and resolved as in IBIS.

- **Output for the task:** Design Object or a part of the final specification that satisfies requirements under consideration.

- **Assumptions** for assessing the applicability of Argument in an Issue.

- **Decision:** Solution of one or more issues (or sub-goals). There may be relationships between a decision and other decisions made earlier as well as decisions generated later.
**Constraint** Explicit linkage between the reasoning processes and the artefact. Constraints can be a formal or informal specification that the system must carry out.

REMAP also uses a Knowledge Representation Language—the Telos language to implement the association between argumentation and design components. Details of this language are out of the scope of the discussion; the central idea is that it is used to support temporal reasoning to infer logical acquisitions and modify and augment a solution. The requirements engineer may refer to a resolved issue or decision in terms of validity time, may write constraints on resolved decisions, may replay the design process and may record changes of requirements and relationships between decisions and the solution. REMAP can be used to capture the complete design history and maintain the one-to-one correspondence between requirements and the design solutions. Indeed, Ramesh and Luqi (1993) develop a support tool for capturing REMAP-structured history of the requirements process in order to improve the traceability of requirements.
In summary, the task-orientation focus and the incorporation of the extended IBIS model with additional components and Knowledge Representation language address the limitation of IBIS in the local argumentation context and its weak connection with design components. However, the REMAP model is based on (and therefore is limited to) the assumption that the requirements engineering process is incremental and cyclic (see section 2.1.4 where each cycle aims at solving a task (goal or subgoal) of the requirements problem.

### 2.2.5 PHI

A very important descendant of IBIS design rationale is PHI (Procedural Hierarchy of Issues), suggested by McCall (1991) to overcome limitations of IBIS. It is based on McCall’s (1986) descriptive theory of design as *issue-serve systems*. It views the design process as a quasi-hierarchy of issues linked by *serve* relationships (McCall, 1986; McCall, 1991). From this point of view, the author argues that there are two types of information missed in IBIS that can serve design effectively. The first type is *non-deliberated issues* which are ignored; indeed, IBIS issues are associated with controversy and deliberation. The second type of missing information is the dependency relationship between resolutions of issues. Both these features are included in PHI. First, it uses a broader definition of the concept of issue: every design question is considered as issue, whether it is deliberated or not. Second, it uses a new principle for the connection of issues that uses the serving and direct serving relationships. If solving an issue (A) aids in solving another issue (B) then the former serves the latter: A serves B. An issue (A) directly serves another issue (B) if and only if the former (A) serves the latter (B) and the former (A) does not serve any other issue (C) which in turn serves the latter
(B). Two alternatives of the serving relationship—antecedents and subissue respectively describe whether an issue is generated earlier or later than the issue it serves. Figure 2.5 illustrates the above in an example of designing a kitchen. As shown in the illustration the IBIS map is a network of issues connected by a variety of different relationships, while the PHI map has a tree-like structure (a lattice tree with shared subissues) with a single root issue—the Prime Issue. The Prime Issue is a design question that produces the resolution of the entire design project.

Figure 2.5: Example of a PHI map (Fischer et al., 1991)

Therefore, the PHI hierarchy of issue represents procedural design knowledge. The author suggests two rules in raising issues: top-down (breaking down an issue to subissues which serve it) and breadth-first (generating all subissues at the same level before generating subissues at a lower level). The primary contribution of PHI lies in providing the designer with an approach to deliberation as well as to decomposition (McCall, 1991). This approach tends to view design process from the top-down perspective (see section 2.1.4). Although deviations from such tree structure are allowed, they are seen as infrequent (McCall, 1991). This procedural process of constructing the hierarchical
structure of issues may be effective for general design, however it may not be effective for requirements problems characterised by ill-structuredness and diversity of viewpoints of different stakeholders and requiring a creative and opportunistic approach to solving them.

The University of Colorado has developed a number of hypermedia systems that link PHI to computer aided design (CAD) tools in order to assist the designers in creating design explanation during the design process, e.g. AAA (Schuler and Smith, 1990), MIKROPLIS and PHIDIAS (McCall et al., 1990), and JANUS (Fischer et al., 1991). AAA (Author’s Argumentation Assistance) supports the creation of argumentation and it is based on a synthesis of PHI hierarchy of design issues and the Toulmin model. MIKROPLIS (microcomputer-based planning information systems) is a hypertext system whose nodes are texts and it is designed to support the navigation of PHI issues. A common weakness of these systems is the lack of connection to the design components. This is addressed in JANUS and PHIDIAS. Although PHIDAS addresses JANUS’s limitations in a number of aspects, for example information retrieval and issue editing and viewing, the essence of these systems is the support for Schön’s (1983) theory which views the design process as a continual alteration of knowing-in-action and reflection-in-action. Indeed, in architectural design, action can be seen as construction and reflection can be seen as argumentation (Fischer et al., 1991). The integration of construction and argumentation is implemented in JANUS and PHIDAS through the combination of a PHI hypertext system with a graphic construction kit. These systems demonstrate that an integrated design and argumentation-based environment overcomes deficiencies of either domain-oriented or argumentation supporting tools used in isolation.
Summary

In summary, IBIS and its descendants have been developed and tested in a wide range of domains and applications. Interest in IBIS-based approaches has also been growing in software development. In requirements engineering, IBIS has been used in eliciting and exploring requirements problems and in managing requirements changes (see section 2.1.5).

Although IBIS is found to be useful in capturing the design history, there are two three problems which need further investigation:

- the use of IBIS in requirements engineering, especially in problem exploring and understanding needs further evaluation. In fact, the research focus has been limited to the context of product (software/requirements) development rather then at the level of conceptual development. Although each sibling in the IBIS family tries to address certain problems in IBIS, most extensions have been limited to the notation itself (i.e. through additional components). In addition, problems identified in different studies are “solved” at the expense of the simplicity, an important characteristic which makes IBIS preferable.

- organisation of information appears to be a common limitation of the IBIS family. Different authors attempt to solve this problem in differently ways. However, neither approach provides an adequate solution to the problem because these authors tend to address the problem only at the representation level of using graphical interface and network of issues, they are not based on a deep understanding of the cognitive process of the problem solving activity.

- although IBIS provides a rich source of process data, and its use in process un-
derstanding and management has not been investigated adequately. Most IBIS and its descendants implicitly assume the incremental development process, for example cyclic process in REMAP, top-down process in PHI and its various implementations. Although PHI explicitly provides the theoretical framework of descriptive design, it is based on assumptions of hierarchical perspective (the top-down approach). Conklin and Yakemovic (1991) believe that gIBIS supports opportunism in design by allowing a flexible construction and connection of issues, however this claim needs further evaluation.

This thesis will examine the use of IBIS and approach its problems at the conceptual level.

2.2.6 Design Space Analysis with QOC

Overview

MacLean et al. (1991a) present an overview of the Design Space Analysis approach and propose QOC as a notation for structuring the design space analysis. They emphasise the need to support the creativity and open-ended characteristic of software design by allowing a number of plausible solutions/designs to be represented and judged. Criticising current support tools as forcing premature commitment of the designer in structuring the design solution, the authors suggest that tools should allow the designer to structure unstructured material initially and later change it gradually. The authors propose the Design Space Analysis approach using the QOC notation and present them through an example. The most popular and comprehensive description and demonstration of QOC can be found in MacLean et al. (1991b).
Design Space Analysis using QOC is a structure-oriented approach to structuring and representing the design space around the design product in terms of the comparison of plausible alternatives and the justification for the artefact. The QOC notation consists of three core components: a design question (Question), its alternative solutions (Options) and a set of Criteria for the assessment of Options. The additional component Argument may be used to justify the assessment of Options against Criteria. The component Argument is seldom illustrated in the QOC diagram, it is often kept in a separate text. Each Question represents a specific part of the design and exploits the design space where possible answers (Options) are being opened up and judged by Criteria that reflect the client’s requirements. A set of Questions represents a global space of the design. So QOC “emphasises the systematic development of a space of design Options structured by Questions” (MacLean et al., 1991b, page 202). Figure 2.6 delineates the QOC notation with a node-and-link diagram.

While QOC and IBIS may share some similarity in their notations, their processes and foci are different. As described above, IBIS documents are recorded overtime as they are
made. QOC documents, however, are often constructed at intervals, typically between phases of the active designing process in order to structure a retrospective design space analysis. Therefore, the QOC approach does not represent the historical record of the design process, it concentrates on the logical design space around an artefact, which can be retrospectively structured by the consideration of its alternatives. The QOC notation supports the post hoc approach to design explanation. QOC documents represent a co-product of design (MacLean et al., 1991b). Advantages of the approach include its simplicity and reusability. The notation is simple. The design knowledge structured using QOC is at a high abstraction level and can be reused in similar projects (Dix et al., 1993). Disadvantages include additional time required for the creation of retrospective analysis (Dix et al., 1993) and the large amount of excessively complex semiformal diagrams.

There have been a growing number of intensive research projects into QOC mainly at Rank Xerox Cambridge EuroPARC (previously known as Rank Xerox Research Centre Cambridge Laboratory), where much of the research has been carried on under the AMODEUS project funded by European Espirit. Research in QOC can be classified into the following main directions: developing the QOC representation and process (MacLean et al., 1991b; MacLean et al., 1991a; Bellotti et al., 1991a; Shum, 1993; MacLean et al., 1993), studying QOC usefulness and usability (Shum and Hammond, 1994; Shum, 1996a; Shum, 1996b; Shum et al., 1997; McKerlie et al., 1993; Jorgensen and Aboulafia, 1995) and incorporating QOC within a specific design method, such as supporting formal specification method using QOC (Johnson, 1996). In most studies, the research method adopted was video-based observation of design sessions where QOC was used as a mechanism for developing and evaluating their design solu-
tion. Most design exercises involve HCI (Human Computer Interaction) exercises, for examples developing user interfaces for hypertext or hypermedia systems, user interface for ATM (Automated Teller Machines), or a support tool for the design of safety critical systems (SAM). The researchers focus on mapping the QOC approach with design practice to produce useful design space representation—a theme suggested by MacLean et al. (1991a). Overall, QOC developers suggest numerous ways of improving and extending the notation. The findings indicate the usefulness of QOC in the construction of design space analysis. QOC has gained popularity and has become a research focus in human-computer interaction and software design. Here I discuss and analyse important research issues into QOC.

Developing the QOC notation and representation

In terms of notation, QOC has not evolved as much as IBIS. There is no significant extension to the core structure of the three primary components Question, Option and Criteria. Most attention has been paid to the representation of the assessment of Option against Criteria. Although the component Argument is designed for this purpose, the weighting of Criteria has been recognised and considered as an important problem in further developing the notation. A few techniques have been proposed to address this problem.

Through an video-based observation of seven designers developing facilities for a hypertext system, Shum (1993) identifies a number of issues regarding to the QOC notation. The author raises the concern about the weighting of Criteria. The assessment of Option which is based merely on a number of supporting and objecting Criteria was found to be inadequate. The author devises Criteria trees to represent inter-Criteria relation-
ships. **Criteria** trees show the hierarchy of general **Criteria** and specific **Criteria** in a design context. **Criteria** which support **Decisions** (accepted **Options**) are presented in bold. **Criteria** trees support the comparison between **Criteria** and play an important role in understanding inter-**Criteria** relationships. Although the author suggests that the **Criteria** tree is “an obvious representation via which **Criterion** weighting could be assigned and modified” (Shum, 1993, page 38), no clear mechanism for doing so has been offered. One conclusion drawn from this study is that the (additional) weighting of **Criteria** is needed and should be an important property of the **Criteria** tree to increase its usefulness.

In his subsequent studies, Shum (1996a) affirms that the prioritisation of **Criteria** is important and suggests that it does not necessarily create overheads as the designer would prioritise **Criteria** only when needed. Observing twelve pairs of designer (including both professional and students) redesigning an ATM (Automated Teller Machine) interface and a Ph.D. student designing a Small-talk data structure, the author analyses and strengthens the argument for **Criteria** weighting. In his view, the strength of **Assessment** link between **Option** and **Criteria**, denoted by the thickness of the line as +/- or neutral, may not be sufficient. The author calls for “a more sensitive **Assessment scheme**” (Shum, 1996a, page 209) to be developed as an extension to the QOC notation. He states that this would not add unnecessary effort as the designer would make use of it only when needed.

Later, reflecting upon the use of QOC as a mechanism to analyse and understand the brainstorming activities of the software designer (see also Chapter 2, section 2.1.4), Khushalani (1997) (a non-QOC developer) suggests using number to prioritise **Criteria**. Unfortunately, there has not been further investigation or evaluation of the technique.
Since then, to my knowledge, there has no other specific suggestion made by QOC developers with regard to this problem.

Similarly to IBIS, there are different ways for representing the QOC information. Shum’s (1993) study is one of the most comprehensive studies which examines the representation issue. Observing seven designers retrieving QOC-structured information in response to design queries for a period of an hour, the author concludes that multiple representation forms are needed for different kinds of queries. Graphs are preferable because they support easy navigation and representation, especially of substructures and multiple relationships between Options and Criteria. Lists (indented texts) are useful only for well specified search tasks due to their limited number and clarity of indentations on a page. The author also acknowledges other representation forms, such as hypertexts for representing the table of content view and matrices for representing the large scale of trade-offs, particularly when assessing a large number of Options against a large number of Criteria. Nevertheless, graphs and lists still remain the most popular representation forms throughout most empirical studies into QOC.

**Developing the design space using QOC**

Research into the development of the design space using QOC is intensive. The primary research areas include: heuristics for design space through formulating and structuring QOC components, the design process model incorporating QOC and the cognitive dimension of creating and using QOC with the evolution of QOC documents during the design process.

Different heuristics for developing design space through formatting and structuring QOC components can be found in early work of QOC developers (MacLean et al., 1991b;
Bellotti et al., 1991b; MacLean et al., 1993). MacLean et al. (1991b) present general heuristics for creating a design space analysis using QOC. They include “local” and “global” heuristics to build design space. The local heuristics concern primarily how to generate and evaluate argumentation components locally within a Question, namely: use Questions to generate Options, consider distinctive Options, represent both negative and positive Criteria, overcome negative but maintain positive Criteria, use Options to generate further Questions. The global heuristics aim at broader design issues, such as the modularisation and coherence of the design. They are: identify Options that generate dependencies, look for novel combinations of Options, design according to a set of Criteria, and search for generic Questions. The dependencies between Options are seen as important in decomposing the problem and defining modules in the top-down design approach. Generic Questions are also considered to be essential in determining the overall coherence of the design.

Bellotti et al. (1991b) also stress the importance of having good design questions. They argue that the design questions structure the way the designer views the problem and where he/she looks for solutions. The authors describe a number of heuristics for generating and formulating good QOC Questions. These heuristics are demonstrated through an example of designing an ATM (Bellotti et al., 1991a). Briefly, the heuristics include: focus on a single concern, make important assumptions explicit, use assumptions to generate further Questions, maintain an appropriate level of abstraction across different Options of a Question, and have all Options of a Question address the same Question. Clearly at this stage, the heuristics proposed by the various authors tend to support the systematic and wide exploration of design space.

Later, MacLean et al.’s (1991b) heuristics are reviewed and added into a design process
model (MacLean et al., 1993) (see below for details). In this work, the authors describe the heuristics as declarative and suggest that they can be used at any phase of design. According to the authors, these heuristics, used in the declarative manner, can “provide leverage or ‘cognitive scaffolding’ for reasoning within a design space” (MacLean et al., 1993, page 201). Although their primary role lies in supporting the systematic development of the design space, the heuristics can “provide a mechanism for coping with the opportunistic aspects” which are characteristic of design behaviour (MacLean et al., 1993). This claim may open a new insight into the capability of QOC, however, it is rather simplistic and needs to be explored and examined adequately.

Having developed the QOC heuristics, MacLean et al. (1993) incorporate them into the design process. The authors propose a sophisticated design process model to support systematic development of a design space representation using QOC. The process model consists of two components: five phases of the design process and heuristics for structuring QOC documents. The first component of the process model consists of five phases with QOC activities recommended for each phase. The phases are: identifying relevant information, structuring material into rough QOC, fleshing out design space, reformulating design space and making decisions. Although the authors recommend the sequential order of the phases, they also add that the order may be flexible and not necessarily strict. The second component is declarative and is based on heuristics for creating QOC documents which are first described in MacLean et al. (1991b) (see above). This process model is demonstrated with an example of developing the design space of the interface of a distributed multi-media CSCW system. The authors emphasise that the model provides only a framework and notation for developing and representing design space analysis and admit that it does not intend to describe a com-
plete design space. The authors offer a general suggestion that QOC should be used where it might be beneficial, in situations such as in solving poorly understood or critical issues, in reaching agreement between developers and in clarifying a solution. A more detailed analysis of when to create QOC would be beneficial to the designer in order to make use of the process model. This is very important because according to the first component of the model, the model may lead the designer to use QOC in an *ad hoc* manner, *i.e.* in actual design sessions. Indeed, QOC, used in actual design sessions, can be placed in the centre between prescriptive and descriptive ends of the continuum of ways in which design rationale can be used (Shum, 1994).

In addition, the structuring of *Criteria* trees (see above) for representing inter-relationships between *Criteria* can also be considered as an additional part of developing the QOC design space. Indeed, *Criteria* trees provide the accessibility and coherence of the QOC base (Shum, 1993). The author points out that this is where IBIS and QOC are different. He strongly asserts that the coherence of the QOC structured documents is critical to its accessibility and reusability while reusability is not a particular objective of IBIS (Conklin and Yakemovic, 1991). In his view, the structure of QOC documents provides “*the logical content and structure of reasoning*” (Shum, 1993, page 40) and these may outweigh the loss of the narrative characteristic of IBIS. In fact, the usefulness of the generation and evolution of *Criteria* in guiding the design activity is confirmed later in MacLean and McKeirle’s (1995) study.

The most comprehensive analysis of cognitive tasks involved in ‘authoring’ and representing design space analysis is offered first by Shum (1991b) and Shum (1991a) and later by Shum (1996a), Shum (1996b) and Shum et al. (1997). In these studies, the author argues for the importance of understanding the relationship between designing
and the structuring of QOC. Indeed, he asserts that an understanding of the cognitive tasks involved in structuring QOC is fundamental in developing tools for authoring and retrieving the QOC information. The authors identify four cognitive dimensions: role expressiveness, repetitiousness and knock-on viscosity (the systems’ resistance to change), hidden dependencies and premature commitment. He promotes the need for mapping between concepts the designer works on and the vocabularies of a design explanation notation. He also emphasises the importance of representing hierarchies of issues and criteria as well as the dependency relations between decisions and consequent design explanation. Recognising the danger of premature commitment of the designer in structuring design space, he stresses that this can lead to poor design questions because early questions may not address the real issues. Moreover, he argues, a coherent organisation and reanalysis of design explanation is needed for creating and updating design explanation and promoting its reusability. The author proposes an approach to supporting evolving design space which reflects “iterations through intermediate representations of the design space as new issues and perspectives are uncovered” (Shum, 1991a, page 245). In this approach, design explanation evolves from a rough state to a coherent state through initially recording and labelling reasoning ideas, and gradually updating and restructuring the design explanation.

To explain this use of QOC, Shum (1991a) locates different design explanation approaches within a two dimensional space of when the rationale is created (i.e. from real time to after original deliberation) versus the kind of rationale (i.e. from narrative to rationalised information) (see Figure 2.7). According to the author, IBIS is close to the narrative information in real time with little or no revision at all, whereas QOC (used as suggested in the above approach "covers virtually the whole temporal dimension"
(Shum, 1991a, page 264) and supports the revision and refinement of the QOC representation. Although acknowledging that initially, the creation of rough QOC would be similar to using IBIS, the author stresses that QOC represents the abstracted rationale and does not capture the original design process.

The idea of an evolving QOC analysis from the ‘rough’ state to the coherent argumentation is confirmed in a longitudinal study where QOC was used in developing a Safety Argument Manager (SAM) over three years involving seven designers (Shum et al., 1993). In this project, QOC was used to organise the project documents and summarise the development and current state of the project through developing a design space. The authors confirm that the “hybrid approach adopted here rationalised established, more stable knowledge, but tracked ongoing discussions in a more process-oriented manner (with, however, a view to subsequent rationalisation)” (Shum et al., 1993, page 44). Therefore, they strengthen the argument for supporting the ongoing process as well as
allowing more stable and concise representation of argumentation.

This QOC approach is also examined in a series of further empirical studies into understanding the relationship between designing and structuring QOC (Shum, 1996a; Shum et al., 1997). The evolution of the QOC documents is confirmed to be beneficial. The rough QOC serves the design team as a short term working memory whereas the ‘final’ rigorous and coherent QOC provides a long term memory resource (Shum et al., 1997). These authors suggest that the rigorous and coherent QOC is reusable and comprehensible to other designers. The studies also gain a deeper understanding of the QOC cognitive process and reveal a number of issues in using QOC. A preliminary summary of the findings can also be found in Shum (1994).

The cognitive tasks identified from these studies include: classifying ideas into the QOC components, naming and labelling the components, and structuring and restructuring the QOC documents. The authors judge that the articulation of useful Questions, Options and Criteria is difficult and involves a number of cycles of representation and evaluation and switching between different parts of the design structures. They report that the designer moves opportunistically between different Questions and Options in different problem areas, or moves to discuss Questions and Criteria without discussing Options in detail. They conclude that the construction of QOC is opportunistic, not smooth and top-down, even when the decisions and arguments are known. Their observation shows that “the process of developing QOC analyses is quite different from the orderly structure of the final product” (Shum, 1996a, page 189). The analysis of the restructuring of QOC components is limited only to the context of QOC components, for examples the breaking down of a Question into separate Questions, making explicit design assumptions previously embedded in a Question. It lacks elements of the context
of the design product and the cognitive design process. Therefore, the relationship between the structuring of QOC analyses and the design process has been reported inadequately: what happens to the design product when the designer restructures his/her rationale?

The authors also highlight difficulties in using QOC components. QOC Question is ‘designed’ to be generic and to represent general dimensions of the design space and therefore does not offer analytical power to the designer. The designer’s activity is described as gradual refinement of the design structure (Shum et al., 1997), however, QOC does not support the expression of Option evolution. In addition, the evidence shows that Criteria are ‘useful only at a global level of application’, rather than at a local level of specific problems within the design space (Shum, 1996a, page 202). Further, the authors argue and conclude that QOC analyses, focused on the examination of competing Options, are inappropriate for the depth-first design approach. Thus, they suggest adopting QOC when the design problem demands the breadth-first design approach.

These conclusions are different from the wide exploration and systematic development of the design space, described and recommended in the early work by MacLean et al. (1993) and Bellotti et al. (1991a) (see above). The difference may be explained partly by the difference in their uses of QOC. MacLean et al. (1993) and Bellotti et al. (1991a) suggest using QOC in a more post hoc manner while Shum’s (1996a) and Shum et al.’s (1997) studies involve an evolution of QOC documents from the initial ‘rough’ state towards the rigorous and coherent state as time progressed. Therefore, the ‘rough’ QOC documents and their evolution might capture the history of the rationale made by the designer that the truly post hoc QOC, created in the previous studies, might not reveal. The opportunism in representing the rationale using QOC even when decisions
and arguments are known might indicate that the designer does not stop engaging in
the problem understanding and solving activity when creating post hoc QOC.

In addition, QOC is found to be supportive when dealing with poorly understood design
spaces and to be obstructive when evaluating well-elaborated design spaces (Shum,
1996a; Shum et al., 1997). This finding is consistent with, and further clarifies the
previous suggestion by MacLean et al. (1993) (see above) that QOC does not intend to
provide a full description of design space. Shum et al. (1997) go further in an attempt to
identify the ‘boundary condition’ indicating to the designer when to apply QOC. They
analyse their empirical data, discuss the idea of ‘wicked problems’ identified by Rittel
and Webber (1973) and argue that “degree of wickedness” depends on the designer’s
expertise and the design stage they are at in the design progress. The authors suggest
that the designer may rely on his expertise to determine kinds of problems and create
QOC at points when it can offer analytical leverage. This suggestion is consistent
with the early work by MacLean et al. (1993) and reflects what happens in practice.
However, its shortcoming lies in the reliance on using personal expertise. It would be
very difficult for a designated scriber to determine when to construct QOC, especially
as this often happens in QOC case studies and in common design rationale practice (for
examples, see Shum et al., 1997; Shum, 1994). Moreover, it would also be difficult for
a novice designer to learn and master QOC. Therefore, further studies are desirable.
We will return to this issue discussion in Chapters 5 and 6.

**Incorporating QOC into other design representations**

QOC has been incorporated into various design representations and methods. These
studies are discussed below.
(MacLean and McKerlie, 1995) study the use of QOC in designing a user interface for educational hypermedia systems over twelve months. One member of the design team had the additional responsibility of documenting design issues and decisions using QOC. The design representations used in this project were relatively informal and include tasks, story boards and scenarios. QOC was used as a central representation to coordinate these use-oriented representations of the design. The QOC documents were used to organise and index these representations. The authors analyse the relationship between QOC representation and the use-oriented design presentations. They conclude that QOC is useful, it supports design representations by abstracting away from these concrete design representations and by summarising and generalising their key attributes. They suggest a complementary use of QOC and systematic (use) design representations.

Bellotti et al. (1995) report findings from the six year project EMODEUS which aims at integrating different HCI techniques of system modelling, cognitive user-modelling, interaction modelling and design and integrational frameworks. Design Space Analysis is an integrated component of the last technique. The authors refer to QOC used in this manner as ‘modelling-enriched QOC’. In this study, this specific form of QOC was evaluated through three video recorded sessions: a one day workshop and a presentation of QOC analyses followed by a three hour meeting. During these activities, QOC was used to summarise modelling analyses and communicate them between members of the design team. In general, the authors confirm that this multidisciplinary integration of various HCI techniques is a powerful way for managing both technical and cognitive complexities of user interface technologies. With regard to design explanation, modelling-enriched QOC was found to be useful in analysing modelling issues,
communicating them, and providing an accessible and intelligible information source to the end-user. However, the authors highlight the lack of context as a weakness in the approach. They state that supplementary detail is needed for QOC to be accessible outside the design team.

Johnson (1996), a non-QOC developer, proposes Literate Specification method which incorporates QOC into Petri Nets, a formal requirements engineering method. In Literate Specification, QOC is linked to the abstract clauses of the formal specification. The method is illustrated through an application of specifying requirements for a chlorine recovery plant. It shows that QOC could be used to overcome the limitation of formal specification. It assists in communication, provides the status information, represents design arguments and alternatives, and justifies the specification, especially in developing large scale safety critical projects. However, weaknesses of QOC, as identified in the study, include the lack of precision and the difficulty of consistency checking. To overcome these weaknesses, the author develops a CASE tool which links each Option to a formal clause using hypertext and uses a temporal logic interpreter in order to check inconsistency in the QOC documents.

The experience gained from this study confirms the feasibility and benefits of the incorporation of QOC within a specific requirements engineering method. The study is limited to using design explanation in order to help the client/stakeholder understand the product, with its formality and precision. However, the precision is only one of many benefits which formal methods offer. For example, FOOM developers strongly argue that benefits of formal methods also include detecting inconsistency and contradictions, prompting insights into problems and allowing the specification to evolve, thus supporting the process of specification. The incorporated use of QOC within

89
formal methods should be studied in more detail to see its full potential benefits in understanding the evolution of requirements during the specification process.

Therefore, these studies, reporting benefits of incorporating design explanation within specific design environments, HCI and requirements engineering, encouraged me in studying the potential benefits of incorporating design explanation within requirements engineering in an information systems context, specifically within the FOOM method (described in the next section). First, FOOM, a requirements engineering method, based on a synthesis of different approaches, includes a number of different representation formats (use-oriented design representations). Therefore, it would allow me to study and examine the complementary use of a design explanation notation with other representations in a more systematic way. Second, the use of design explanation within a formal requirements engineering method would be analysed from both the product and process perspectives.

**Empirical evaluation of QOC**

Design Space Analysis using QOC has been assessed by various researchers mainly for the purpose of further extending the approach. In general, as described above, QOC is shown to be beneficial. In addition, QOC is also used by its developers purely for the purpose of evaluation, specifically in McKerlie et al.’s (1993) study. These authors report their experience with QOC through an expert evaluation of hyper-systems (hypertext and hypermedia) in a computer based learning environment. They emphasis the usefulness of retrospective Design Space Analysis using QOC in making important design consideration explicit, particularly in examining and presenting possible solutions within a single system (intra-design analysis) as well as in comparing solutions
used in different systems (inter-design analysis).

The QOC notation is also evaluated in a number of empirical studies by researchers, other than the QOC developers (Jorgensen and Aboulafia, 1995; Sutcliff and Ryan, 1997). There are a few findings consistent with QOC developers’ expectations; however, these studies also raise a number of controversial issues around the usefulness and usability of the notation.

Jorgensen and Aboulafia’s (1995) study involves seven groups of twenty three graduate students over five weeks training and two weeks working on an assignment. The design task was to develop use interface for a city tourist map. The data collection method adopted was questionnaires and experience reports by students. Sutcliff and Ryan’s (1997) study involves twelve designers working on a realistic safety critical problem for 15 minutes after a training session. The design session was audio taped. Empirical data include a transcript of the audio tape, a copy of QOC diagrams created by the designers, a feedback report by the designers eight months later. It seems that in the first study, the designers recorded both initially rough and evolving QOC. *i.e.* not in a purely *post hoc* manner. In the second study QOC is used in a *post hoc* manner, however, it is limited by a very short time period. Both studies show the lack of an analysis of context or situation where QOC was used.

In general, both studies provide evidence that QOC is useful and usable. The second study also indicates that it may be beneficial for requirements engineering (Sutcliff and Ryan, 1997). QOC was found to be useful in facilitating participation and contribution from the designers. However, there are some issues which need further research attention.

First, the designers in both studies experienced problems in handling a large number of
diagrams. The designers from the second study also criticise QOC as time consuming. These conclusions are similar to findings from other studies, (for examples Bellotti et al., 1995; Shum et al., 1997; Darke and Shanks, 1996).

Second, difficulties in identifying and classifying design ideas into the QOC components are reported (Sutcliff and Ryan, 1997). The authors explain these as a weakness of QOC in guiding problem decomposition: “QOC does not guide decomposition as clearly as some top down methods” (Sutcliff and Ryan, 1997, page 154). Although the modelling process is expected to be top down by many authors, this finding suggests that a further understanding of the modelling process is desirable. The authors also advise using QOC with some specific modelling methods to assist in more detailed reasoning. In addition, although this may be consistent with a conclusion drawn from Shum’s (1996a), it might be limited by the short timeframe (15 minutes) which might have not been sufficient for the designers to explore and develop the problem space thoroughly in order to structure a QOC analysis.

Lastly, QOC was found to be useful for the organisation of documentation, but less effective for the on-going argumentation and reflection (Jorgensen and Aboulafia, 1995). The designers from this study suggested using QOC to structure information gathered by other means, i.e. not using QOC up-front to capture “flying thoughts”. However, the designers from the second study found QOC to be useful in recording thought processes. This inconsistency might have resulted from different uses of QOC and different design contexts. Further studies, which can provide a rich understanding of context, are needed to clarify this inconsistency.
Summary

In summary, Design Space Analysis using the QOC notation is a major focus of research into design explanation. The retrospective QOC offers many benefits in understanding the design space in terms of possible structures of designs and their pros and cons. The design deliberation, summarised in a rigorous and coherent QOC, may be accessible and reusable. Perhaps, the most important issues which need further research attention may include:

**How to create QOC** There are two ways of using QOC suggested by different authors. They are still subject to debate:

- having a rough and evolving QOC. Disadvantages of this approach lie in the difficulties in generating and recording ‘flying thoughts’, identified in Jorgensen and Aboulafia’s (1995) study, and the difficulties in handling a large number of QOC diagrams.

- using QOC in a purely *post hoc* manner. Disadvantages of this approach lie in the difficulties of recognising when to create QOC.

**The need to study both the QOC and the design processes simultaneously**

The opportunistic characteristic of structuring and restructuring QOC suggests the need for further investigation into the evolution of the rationale information in the context of the evolution of requirements model at the conceptual level of problem understanding and solving.

**The need to incorporate QOC with a design method** Benefits of QOC in providing an explicit design space analysis, the lack of formality (Johnson, 1996) and
the lack of a practical guide to more detailed reasoning suggest potential benefits of the incorporation of QOC within a specific design method.

Clearly, a better process model of using the design explanation QOC within a specific design method is desirable. This thesis investigates this issue.

2.2.7 Other design explanation notations

The Toulmin notation

Toulmin (1958) perceives two rival reasoning models: the mathematical and the jurisprudential models. Although the author acknowledges the simplicity of the mathematical model, he contends that it comes at substantial cost. Indeed, criticising the (Aristotelian and other formal) logicians as seeing their world as subject for principles of valid reasoning, Toulmin (1958, page 177) wrote “their “deduction” limits them in practice to the principles of valid analytical reasoning”. The Toulmin notation is developed as a challenge to the dominance of deductive approaches to reasoning, particularly, Aristotelian logic. There are two levels of analysis in the Toulmin notation: the soundness and the strength of arguments (Toulmin et al., 1984).

At the level of soundness of arguments, there are four argumentation components: Claim, Ground, Warrant and Backing. The Claim of an argument is the conclusion, the ‘discovery’ or the ‘destination’ at which the argument arrives. Grounds comprise a fact (or a set of facts) which are agreed not to be disputed and upon which the Claim is based. Warrants are defined as “statements indicating how facts on which we agree are connected to the claim or conclusion now being offered” contained in Claim (Toulmin et al., 1984, page 45). In the natural sciences, Warrants may be laws of nature,
formulæ; in law, they may be statutes and rules; in medicine, they may be diagnostic
descriptions; and so on (Toulmin et al., 1984). Warrants are supported by Backing
or the “generalisations making explicit the body of experience relied on to establish
the trustworthiness of the ways of arguing applied in any particular case” (Toulmin
et al., 1984, page 61). In the natural sciences, Backing is what can be used to validate
a scientific explanation; in law, it may consist of decisions of courts or legislatures;
in medicine, it may be established by reflecting on the relevance of theory (e.g. the
physiological theory), and so on. Backing explains why the assumed Warrant is valid.
It depends on the professional’s training and experience. Warrant can be seen as bridge-
like statements while Backing can be seen as categorical statements that support the
connection between Grounds and Claim.

At the level of the strength of arguments, there are two argumentation components:
Qualifiers and Rebuttals. Qualifiers describes the strength or weakness of the argument.
The authors identify various values for this component: necessarily, certainly, presumably,
in all probability, so far as the evidence goes, for all that we can tell, very likely, very possibly,
maybe, apparently, plausibly and or so it seems. Rebuttals describe the “extraordinary or
exceptional circumstances that might undermine the force” of the supporting Grounds,
Warrants and Backing.

There are a number of links to connect all these components in order to form an
argument. Figure 2.8 is an example of the application of the Toulmin notation.

Toulmin (1958) argues for the uniqueness of argumentation modes of his model. For
example, the rationale depends on the establishment of Warrants which are permitted
and agreed by professionals in the domain of the argument. Warrants authorise the in-
fERENCE Ground-Claim and form a rational support for the Claim. However, the notation
is criticised as being limited in expressing the design space and lacking a mechanism for the assessment of alternatives (different Claims) according to their pros and cons (Lee and Lai, 1991; Shum, 1991a). In design, Toulmin’s notation would support the documentation of the deliberation process in terms of cause and effect, rather than structure a broad design space around the design product in terms of assessing and choosing plausible alternatives. Therefore, it can be classified within the process-oriented category of design explanation notations.

In common with other design explanation notations, arguments in the Toulmin notation are often represented as node-link diagrams. The key difference from other notations lies in the connection between the link Ground-Qualifiers and the node Warrant while most notations do not permit the connection between a node and a link.

In conclusion, as the first semi-formal graphical representation of arguments, the Toulmin model’s argument pattern and graphical representation strongly influenced many other subsequent design explanation notations. Originally, the Toulmin argumentation
model was designed by analogy with jurisprudence. Nowadays, it has been applied to numerous domains, such as legal decision making (Marshall, 1989), linguistics (Botha, 1970) as well as general design (Schuler and Smith, 1990).

From a generic model to the Inquiry Cycle model of design explanation

Potts and Bruns (1988) propose and develop a generic model for recording reasons for design decisions during the software engineering process. Similarly to the IBIS notation, Potts and Bruns’s (1988) model consists of three basic components: Issue, Alternative and Justification where Issue derives Alternatives and the Justification for Alternatives. This model is represented as an argumentation node in documenting the design process. The design process is seen as a network of intermediate Design Artefacts, connected via the argumentation nodes. The approach is different from other design explanation approaches primarily in respect of the direct connection made between Design Artefact and Alternative: Design Artefact is derived from Alternative. Figure 2.9 describes Potts and Bruns’s (1988) generic model for representing design explanation and recording the design process.

Since there is only a summary of arguments documented in a Justification for each Alternative, the Justification component lacks expressiveness and explicitness, and thus provides only weak support for comparing Alternatives. Furthermore, with the direct connection between Alternative and Design Artefact, this generic model tends to focus on the narrative dimension of the design process rather than offering a powerful reasoning mechanism.

The model may be customised to a specific software development method and is demonstrated through an example in hypertext and a Prolog database suitable to the chosen
method (Potts and Bruns, 1988). A number of components needed to be added to the model. The study also shows that the generic model must be refined and modified in order to accommodate a specific design method (Shum, 1991a).

Later, Potts and Bruns’s (1988) generic model was further refined and modified to become the Inquiry Cycle model for describing and supporting discussions during system requirements analysis (Potts and Takahashi, 1993; Potts et al., 1994). The primary problems that the model addresses include communication, agreement and management of change. The model defines a cyclic process of requirements documentation, discussion and evolution (see Figure 2.10). Requirements are documented in informal text, primarily as scenarios. The core discussion structure resembles previous argumentation models and consists of Question, Answer and Reason. The requirements evolution reflects the ultimate effect of a discussion. Changes to requirements are classified into five sophisticated categories: clarification, retraction, refinement, merging and splitting. The classification is later refined. In a later version of Inquiry Cycle, changes
to requirements are classified into: mutation, restriction and editorial. Briefly, Inquiry Cycle is considered as a formal structure for describing discussions about requirements. Potts and Takahashi (1993) also developed a hypertext tool to support Inquiry Cycle requirements analysis.

![Inquiry Cycle model](image)

Figure 2.10: Inquiry Cycle model (Potts et al., 1994)

Potts et al. (1994) offer a quantitative case study. Issues were counted and classified. The study, *inter alia*, confirms the suitability of the model and suggest that the model may not require *more* effort but it may rather refocus the effort of the requirements engineer. This suggestion strengthens benefits of using design explanation. Other authors suggest that design explanation may pay off the extra cost of creating it (Conklin and Yakemovic, 1991; Fischer et al., 1991). This study suggests that documentation may not necessarily create overheads in design. However, the question of how to use design explanation so that it requires refocused rather than extra effort needs to be examined critically, especially since *documentation* is always a ‘bad’ word in systems development practice. In addition, the authors also suggest further research into the transition from scenario-based requirements analysis to Object-Oriented analysis and modelling.
In summary, the generic model and Inquiry cycle are descriptive and can be seen as an approach to process modelling. These models are best classified within the process-oriented category of approaches to design explanation. Although they are not widely used, they raise a number of interesting issues for future research into design explanation:

- Generic models of design explanation are limited in scope. They cannot replace the traditional systematic design methods. They need to be customised to a specific design method. Potts and Bruns’s (1988) study is the first which shows that this is feasible. Potts et al.’s (1994) study further investigates this issue in requirements analysis, however, its shortcoming lies in the lack of a specific requirements engineering method as an environment in which design explanation is applied and examined. This has been taken into account in this thesis which aims to investigate benefits of design explanation within requirements engineering, specifically, within FOOM.

- Non-intrusiveness is an important characteristic of any argumentation approach. Non-intrusiveness is vital (Potts and Bruns, 1988) in allowing the capturing of historical information about the design process and the reason for design decisions “without imposing a major data entry burden on the designer” (Potts, 1989, page 217). Although non-intrusiveness is not mentioned in Potts et al.’s (1994) study, it might be a key issue in understanding the question of refocusing effort discussed above. Non-intrusiveness is, in fact, an influential factor in choosing a specific design explanation notation for the study reported in this thesis (see chapter 4 and 5 for details).

- Having found some benefits of the model, the authors still question what kind
of design history is useful: “whether the explicit design rationale is helpful or intrusive during constructive design” (Potts and Bruns, 1988, page 425). They wonder whether design explanation may be best done in an introspective manner. Indeed, Inquiry Cycle is later described as artefact-based and is suggested to be used in order to review current requirements (Potts et al., 1994). However, the central focus of the model still lies in discussions of specific issues raised from requirements documentation rather than from a global design space analysis.

- These studies are early steps into the integration of an design explanation model within the evolution process of software engineering. The models proposed here are strongly based on an incremental evolutionary process of analysing and changing requirements, where “the analysis progresses towards a more precise specification” (Potts et al., 1994, 23). Since the process of requirements engineering needs to be further examined (see section 2.1), these models may be simplistic and may not be expressive enough in describing the dynamics of the process and the underlying rationale. The use of design explanation needs to be further investigated based on a deep understanding of the design process. This thesis offers a contribution into further developing this idea.

**Conversation-Oriented Environment for Design (COED)**

The COED project (Kaplan, 1990) is developed mainly at the University of Illinois. It is based mainly on the notions of hermeneutics and Speech Act Theory (SAT) (Searle, 1979). Hermeneutics views language as interpretation—information is passed through language, while Speech Act Theory emphasises the equivalence of language and action—people act through language (Winograd and Flores, 1986; Winograd, 1987;
Searle, 1979). From there, Kaplan (1990) views the design process as a conversation-oriented activity—either as a monologue by the designer or as conversations between different designers. COED aims at supporting design through the coordination of these conversations.

With regard to the design explanation, the author adopts Heidegger’s (1967) distinction between the breakdown and throwness concepts to determine what kind of rationale is to be captured in design. Breakdown refers to the moment at which conversations are brought to a halt because hidden problems are revealed and important questions arise. Conscious effort is required to solve these questions. The breakdown moment is crucial in the design process. Although Kaplan (1990) does not mention it in his work, this is consistent with what has been discussed early in section 2.1.4. In contrast to breakdown, throwness refers to the unconscious dealing with information. The author suggests that design decisions happen at breakdown moments. This is also consistent with Schön’s (1983) analysis that reflection is required at breakdown points in knowing-in-action. The COED project is aimed at capturing these breakdown points.

In terms of notation, a conversation is a sequence of utterances which may be either verbal or non-verbal and may be concerned with information gathering, shaping, representing and passing, etc... COED introduces the Conversational Frame Model (CFM) to represent utterances during the design process. There are five categories of utterances in the original SAT: commitment, directive, assertion, declaration and emotive (see Searle (1979) and Winograd (1987) for more details). In COED, other categories can be introduced as needed. Each utterance (CFM) is characterised with the illocutionary force (the weight) and the propositional content (the volume).

The design process can be described as a natural conversation in a specific context,
consisting of a set of utterances, configured in semantic way that lead from one action
to another. The result is a decision of a design problem.

COED is implemented using the hypertext technology. The activity, performed by the
designer, forms a hypertext graph. Each person’s utterance CFM corresponds to a node
in the graph.

Conversation Builder (CB), a support tool, has been implemented under the COED
project in order to provide a flexible and active environment to support collaborative
design activities (Kaplan et al., 1992).

COED serves as a communication mechanism and clerical assistance among all stake-
holders involved in a project. COED records important conversations between develop-
ers, users and implementors. Therefore, COED can improve requirements traceability.
In addition, important moments can be reviewed later.

To sum up, the COED conversation-oriented approach is built on the foundation of
the equivalence of language interpretation and action. It supports capturing design
explanation and playing back breakdown points which occur during the design pro-
cess. Similarly to IBIS, COED focuses on capturing a historical record of the design
process. While the IBIS model supports the decision making activity through high-
lighting possible alternatives and their arguments, COED, based on philosophical and
design theories, supports a graph of thinking process (a mental model) with (brain)
nodes linked together through a number of thoughts (mental actions). Although the
COED-structured information is claimed to be invaluable and reusable in other projects
(Kaplan, 1990), further studies are needed for its empirical evaluation. This is espe-
cially important as the complicated classification of utterances may require extra effort
from the designer.
Evolutionary Object Model (EOM)

EOM provides a set of concepts to structure and to represent the process of requirements engineering (Moreno and Souveyet, 1993; Rolland, 1994). It is developed at the University of Paris. EOM is a decision-oriented process model. It focuses on the decisional aspect of the design process, which leads to the final artefact in a step-by-step or decision-by-decision manner. A tracing model called F3 is developed as a part of EOM in order to record the history and explanation of the design process.

In EOM, the design process is described as a network of evolutionary objects (EO) (Moreno and Souveyet, 1993). EOM is an object-oriented model. EOM objects can be at three levels of abstraction: meta-model, model and description (see Figure 2.11). The Product meta-model defines EO types independently of any design methodology. The Product model is comprised of a set of EO classes; each of them being an instance of an EO type from the meta-model depend on a specific design methodology chosen for use. This model records the evolution of EO in terms of a set of complicated concepts of inner, spatial and temporal descriptions (for details, see Moreno and Souveyet, 1993; Rolland, 1994). The Product description is the artefact itself. It contains instances of EO Classes.

![Figure 2.11: Abstraction levels in EOM (Moreno and Souveyet, 1993)](image)
The evolution of an object is documented in the form of successive versions. Each version is annotated with a timestamp and the transformation that creates it. Each transformation is called as an atomic step and is recorded as an Action (a selected Decision that performs the transformation) taken in a Context (a current situation of the design product) and considered Decisions (possible options) (see Figure 2.12). Therefore, the design process is represented in the form of a graph of actions. Each action is an aggregation of action-name, effective parameters (being modified parts) with names and new values, and a context reference.

![Figure 2.12: Basic model of development loop (Moreno and Souveyet, 1993)](image)

In conclusion, EOM (and the tracing model F3) can be classified into the process-oriented design explanation category. This approach supports requirements engineering through recording the history of the evolutionary design product rather than capturing the rationale behind it. Therefore, its primary contribution lies in recording the mutation of the design and improving requirements traceability. However, this approach does not support the reflective nature of design. In fact, little is recorded to support the problem understanding and decision-making activities and to explain why the final product is the way it is.
Decision Representation Language (DRL) and SIBYL

In the spirit of a Greek mythological character who gave wise advice to people, SIBYL was “born” to assist the designer in decision making (Lee, 1990). Similarly to QOC, SIBYL can be classified into the structure-oriented design explanation category. However, SIBYL is based on a developed language, called Decision Representation Language (DRL). DRL is proposed and developed by Lee and Lai (1991) using their systematic framework for the evaluation of design explanation presentations. DRL qualitatively structures the design deliberation and the shared knowledge more explicitly than other design rationale notations. DRL is claimed to be the most expressive language for the representation of design explanation (Lee and Lai, 1991).

In terms of notation, DRL constituents are based on the intuitive model of reasoning which is similar to other notations. It consists of a Decision Problem DP (question) which is being considered; Alternatives A (option) which are investigated; and a set of supporting or opposing Claims C (Criteria) for the assessment of Claims. In contrast to other simple notations, DRL has a rich set of objects and grammar to manage the dependencies among the objects. There is a set of goals and subgoals (in specific local contexts): G (goal/subgoals) to be achieved when solving a design problem DP. Claims are linked to Goals via the relationships Achieves (A,G) or Is-an-alternative-for (A,DP). The central activity of the assessment of the plausibility of the Achievement (A,G) and the importance of the goal G. There are a various degrees for the assessment of Claims, such as: Support, Deny, Presuppose, Influence and Query. The components Question, Procedure and Group are designed to support the relation between Claims and Procedures. Instead of the simple pros and cons examination of design options in QOC, there are a number of sophisticated levels for the assessment of the importance of
goals/subgoals \( G \) and a possible solution \( A \): High (H), Medium (M), Low (L) and unresolved.

The syntax is summarised in Figure 2.13.

![Figure 2.13: DRL vocabulary (Lee, 1990)](image)

In terms of representation, the rationalisation may be represented in the graph (Figure 2.13) or table forms (Lee, 1990). With regard to the table form, SIBYL uses the two dimensions Decision Matrix of Goals and proposed Alternatives. Values of the cells of the Matrix may be H, M, L and unsolved. These values can be assigned and updated from time to time. The underlying DRL semantic rule assists the designer in structuring a logical space, in exploring competing views and in shaping a consensus solution.

Therefore, DRL, implemented in SIBYL, supports monitoring decision dependency, precedent management (decisions sharing the same goal), viewpoint management (claims sharing common features (Group)) and plausibility management (through various degrees for the assessment of alternatives) (Shum, 1991a).
In SIBYL, the design knowledge and different viewpoints are shared by spreading the current Decision Matrix’s data in messages or files. These data can be retrieved to recall past design decisions or to review attitudes (Lee, 1990).

In conclusion, DRL is a formal and generative structure of thinking. SIBYL has a rich language with a rich vocabulary of argumentation components with semantic interrelationships among them. Being based on DRL, the SIBYL system supports a more qualitative and intentional assessment of design decisions than other design explanation support systems. This design explanation approach is characterised with two controlling ideas of thorough semantic reasoning and knowledge sharing. However, this approach is complicated and requires significant overheads of classifying and structuring rationale ideas into the argumentation components.

Meta-Argumentation Workbench (MAW)

MAW, developed by Sargeant (1993) and Shanks et al. (1994), is based on a synthesis of various design explanation models. This model is capable of creating and representing the deliberation information by different notations during the design process.

Based on the Entity-Relationship modelling technique, the model enables the use of Node and Link entities to represent argumentation components of different notations. The common structure of various design explanation models can be generalised as a node-link diagram (Sargeant, 1993; Shanks et al., 1994). In the Meta-Argumentation Model, nodes and links of various argumentation models are classified in high generic entity types as Node and Link types which are subtypes of Argumentation Component Entity Type. For example, Issue, Position and Argument are three Node types and Responds to, Supports, Objects-to and Is-SubIssue... are Link types in instanti-
ating the gIBIS notation. Similarly, Question, Option and Criteria are Node types, and Generates, Responds-to, Is-positive-assessment-of and Is-negative-assessment-of are among Link types in generating the QOC notation. The semantic rules which define the ‘legal’ components and allowable connections between Link and Node are stored in Binary Link Type Rule and Binary Link Type Rule Type. Figure 2.14 shows the Meta-Argumentation Model. The Meta-Argumentation Workbench was implemented to demonstrate the feasibility of the model. There are two modes of operation: the initialisation and the creation of design rationale. In the initialisation mode, the designer retrieves a given notation from a parameter file. This file configures all the selected notation’s components and their allowable connections. In the creation mode, the designer works with the actualised argumentation notation.

In this way, the model captures the essential of most pre-existing design explanation models and is capable of actualising and switching between them. It is very important to note that MAW should not be seen as a design explanation approach for it does not guide what, when or how to capture and record the rationale behind the design. Its
contribution to design explanation lies in the provision of flexibility in using various
design explanation notations.

In addition, the workbench raised a number of issues, for example how to support
multi-users in collaborative work and how to implement n-ary links between argumenta-
tion components (Shanks et al., 1994). The most important issue is the need for the
integration between design explanation and design objects in order to promote trace-
ability and to improve understandability of the design artefact (Shanks et al., 1994).

In this thesis, the specification of requirements for the enhanced tool becomes the re-
quirements engineering project under study. Further, the outcome of the thesis confirms
the need for developing such a meta-argumentation model and the integration of the
model into the evolution of requirements.

2.2.8 Summary and conclusion

There are indications from the literature that design explanation may offer designers
many benefits, such as:

- improvement of decision making. Design rationale may raise crucial issues, ratio-
nalise discussions and avoid missing important considerations about requirements
  by each designer as well as by a group of designers.

- reuse of rationale arguments in similar situations and related projects. This may
  avoid solving the same problem of making the same decisions over and over again
  as the specification evolves.

- explanation of a specification. It is important to understand a specification in
  terms of its relationship to possible alternatives and/or other similar projects.
• assistance in communication between various participants including stake holders, users, specifiers and other developers.

Nevertheless, Shum and Hammond (1994), in their literature survey, show that there is little observational, experimental evidence which supports the claims of utility and usability of design explanation. Indeed, most evidence is still rather general or anecdotal. Moreover, Moran and Carroll (1996) state that design explanation is still in an early stage. Indeed, the above discussion highlights a number of significant issues (see earlier) which need to be addressed in future research into design explanation. The above discussion also shows that most research into design explanation has been carried on in the field of HCI rather than requirements engineering. Undoubtedly, the use of design explanation within requirements engineering needs to be explored and studied in detail.

To conclude this section, let me quote what has been said about directions for future research into design explanation: “the context in which a notation is used will determine the expertise present, and the time available to structure deliberation—these “what, when and how” factors cannot be ignored when designing a notation” (Shum and Hammond, 1994, page 640). Building upon current approaches and notations and understanding of design explanation, this thesis offers a further understanding of “what, when and how” of design explanation when being used in requirements engineering.

2.3 FOOM

Previous studies indicate the need for exploring the usefulness of design explanation (and to incorporate it) within a specific design method (Potts and Bruns, 1988; Fischer
et al., 1991; MacLean and McKerlie, 1995; Bellotti et al., 1995; Johnson, 1996). Formal Object-Oriented Method (FOOM) (Swatman and Swatman, 1992; Swatman, 1996; Fowler, 1996) was chosen as the host requirements engineering method. This section introduces FOOM briefly and justifies the choice of FOOM as the requirements method under study.

2.3.1 An introduction to Formal Object-Oriented Method

FOOM is based on three theoretical pillars:

- the object orientation theory in which requirements are modelled as a set of interacting, encapsulated objects, each object belonging to a class. The object-orientation theory provides many benefits, such as modularity, abstraction, classification, encapsulation, polymorphism, robustness and reuse (see for examples, Henderson-Sellers, 1997; Henderson-Sellers and Edwards, 1994; Booch, 1994). In FOOM, the object-orientation theory is instantiated by a modified object-oriented notation, MOSES (Henderson-Sellers and Edwards, 1994), a predecessor to OML.

- the mathematical specification language Object-Z which provides precision, and therefore may offer deep insights into the problem being specified, moreover, it also supports the object-oriented structure of specifications,

- a socio-organisational theory based on Checkland and Scholes’s (1990) Soft Systems Methodology which promotes open different subjective interpretations by various people involved in the development of a specification and addresses the social context within which the system will be used.
The FOOM approach also includes a number of structure and process components, techniques and recommended activities (Fowler and Swatman, 1997).

According to Fowler and Swatman (1997) the FOOM process model is described as cyclic. Figure 2.15 shows the FOOM modelling process. Each cycle is composed from information analysis, modelling and validation. During the information analysis process, the clients’ requirements are interpreted, analysed and structured. During the modelling process, in which requirements models are constructed, informal, semi-formal and formal modelling techniques are used in parallel (see Figure 2.15). During the validation process, the results from the modelling process are evaluated and conflicts identified from the formal modelling are solved by client-validators.

Figure 2.15: FOOM modelling process (Fowler and Swatman, 1996)

The FOOM deliverables are a number of specification documents in three tiers:

- an executive summary,
- Event Chain diagrams with some Object-Z fragments to facilitate the validation of system behaviours,
• modified MOSES Object/Class diagrams, Inter-Object Communication diagrams with complete Object-Z and textual documents.

The first document, an executive summary, is prepared for managers in the form of a very high level overview of the functionality of the system in textual form, supplemented by informal diagrams. The second document, Event Chain diagrams linked to fragments of the formal specification, represents the behaviour of the system. This section of the specification is used to explain the behaviour that is defined by the specification to the clients responsible for accepting it (validators). The third document, a complete Object-Z specification is provided with explanatory material, including static and dynamic object-oriented notations adapted from MOSES. The last section of the document is aimed at designers and implementors. The detailed description of these diagrams and the semantic mapping between them can be found in Fowler (1996), Fowler et al. (1995) and Wafula and Swatman (1996).

In order to read Chapters 4, 5 and 6, it is sufficient to understand that a FOOM specification includes an informal textual explanation; object-oriented structure, communication and Event Chains diagrams; and a formal Object-Z model. A more comprehensive overview of FOOM will be included in appendix A.

2.3.2 Why FOOM?

Based on a synthesis of socio-organisational contextual analysis (Checkland and Scholes, 1990), Object-Oriented approach MOSES (Henderson-Sellers and Edwards, 1994) and formal specification language Object-Z (Duke and Rose, 1995), FOOM reflects an Information Systems methodological context and offers a formal specification method. In investigating source requirements which may be described as inherently vague and
ambiguous, FOOM offers us the following advantages:

- Being grounded on various theoretical foundations ranging from soft to hybrid perspectives, FOOM offers the researcher an open opportunity to examine problems/issues as appropriate.

- The formal and Object-Oriented models specify requirements with discrete design objects (components) using the Object-Oriented concepts, such as object, class, method and Object-Z logical statements. The complexity of FOOM models can be measured in a variety of ways.

- The formal FOOM models precisely reflect and represent our understanding of the requirements problem at different stages of development. Changes made to requirements models are observable, specific and precise. Therefore, the researcher is able to follow up our learning about the problem and to examine how the requirements model developed incrementally.

- The variety in FOOM models offers us an opportunity to learn whether and how structured documentation of rationale information could provide a unified view across different views on requirements.

The choice of FOOM does not significantly affect the generalisability of the research outcome. Requirements engineering methods vary, but they all focus on one or more dynamically changing “models” of the problem context under study. Therefore, the researcher strongly believes the research outcome can also be extended to other requirements engineering methods. However, by revealing a new understanding of the dynamic process of requirements engineering (see Chapter 6), this thesis raises a question of the appropriateness of this and other requirements engineering methods as they,
in general, are based on the current understanding of an incremental evolution of requirements, rather than offering a directly applicable conclusion. Chapters 6 and 7 will discuss this in more detail.

2.4 Synthesising views

Section 2.1 discusses various issues in requirements engineering and leads to the following conclusions:

- Current understanding of the process of requirements engineering embodied in most current methodologies, approaches and techniques is insufficient. A further understanding of the process is highly desirable.

- The requirements engineering process is dynamic. Intermediate states of requirements and the knowledge of how and why requirements decisions are made might be critical in understanding the process and the cognitive behaviour of the requirements engineer.

Section 2.2 reviews the history of development and analyses the state-of-art issues in design explanation. This section leads to the following conclusions:

- There are predictions that design explanation might be useful in supporting the process of requirements engineering.

- Although design explanation is useful, it comes at a cost. The questions of what kind of design explanation is useful as well as when and how to create design explanation are still poorly understood and are a debatable topic.
The above discussions (especially of the first part) suggest that systematic documentation of the requirements model and underlying arguments may be useful in understanding the evolution of requirements and in monitoring and improving the requirements engineering process. The discussions also encourage the researcher to explore the use of design explanation within requirements engineering though incorporating it within a specific requirements engineering method. Section 2.3 briefly describes FOOM as a chosen method under study and justifies the choice. The next chapter will formally formulise research questions and select an appropriate research approach for the study.
Chapter 3

Research Methodology and Research Design

3.1 Research objectives

The purpose of this thesis is to investigate the use of design explanation within Requirements Engineering. The investigation is based on reflection upon the use of design explanation within an actual application of FOOM to specify requirements for a prototype CASE tool. The CASE tool will be designed to support requirements engineering using FOOM. The specific objective, which is addressed and studied in-depth in this thesis, is to investigate if and how design explanation can be incorporated within Requirements Engineering to explain design decisions and support the traceability and understandability, and hence, to improve the quality of the requirements engineering process and specifications. Beside this, the general intention is to strengthen findings of previous studies into FOOM and further evaluate the method in order to extend it.
During the requirements engineering process, the customer’s requirements, which are often ambiguous, incomplete and may be contradictory, are analysed (may be changed) and transformed into an accurate, complete and precise specification to which the developed system must conform. A complete FOOM specification is a network of specifications in the form of text, semi-formal Object-Oriented diagrams and the formal specification language Object-Z (Fowler and Swatman, 1996; Fowler, 1996; Swatman, 1996).

During the requirements engineering process, a large number of decisions are made, each of which may result in a change to the model of requirements being built. This process, thus, is often very complicated and needs to be monitored. Previous studies indicated that design rationale could be beneficially used to support requirements acquisition (Potts et al., 1994; Ramesh and Dhar, 1992; Yakemovic and Conklin, 1990). The two research questions this thesis addresses are:

- **Is design explanation useful for the description, explanation and hence the “management” of the evolutionary process of a specification?**

  A previous study by Fowler (1996) found that during the modelling process requirements undergo a number of iterative changes between the informal and semi-formal forms towards the semi-formal and formal forms. The final specification should be precise and accurate, and most importantly, it must conform to the clients’ informal requirements. My proposition is that the use of design explanation will increase confidence in the conformance, the consistency and appropriateness of the specification.

- **If design explanation can be shown to be useful, how should it be incorporated within FOOM?**
Fischer et al. (1991) argued and demonstrated that an integrated design and argumentation-based environment overcomes deficiencies of either sole domain-oriented or argumentation supporting tools. Most previous studies (MacLean et al., 1991b; Conklin and Yakemovic, 1991; Toulmin, 1958; Kaplan, 1990; Kaplan et al., 1992) into design rationale are limited to propose and develop, either ad hoc or post hoc, notations and CASE tools to capture design decisions and deliberations. However, this thesis shifts the focus of study from the use of a prescribed notation for general requirements analysis to the use of a selected but not restricted design explanation approach and its possible adaptation to suit and improve a specific requirement engineering method, namely FOOM.

Thus the project is concerned with extending a method through observation and qualitative analysis of its application in a real case. It is worth noting that though these two research questions, especially the second question, are helpful in preventing overwhelming large volume of data to be collected, they both are still tentative and no answer is guaranteed a place in the research outcome. The research is best characterised as investigatory.

There are a number of research approaches which could be used for the project. This chapter explains and justifies the research approach adopted for the project, namely action research. This chapter will:

- discuss some issues considered when choosing an appropriate research approach for the project. The structure of the discussion is set as follows:

  - overview of positivism and interpretivism as two research paradigms used within Information Systems,
– discussion of some critical issues when choosing a research approach,
– description and discussion of case study and action research as two possible research approaches to be adopted.

> • justify the chosen research approach and present the research design adopted for the project. The structure of the description is as follows:

– description of the context of a programme to develop FOOM, and the position of the project in this programme,
– justification of the adoption of the action research approach for the project and explain how the action research approach is adapted to suit the project,
– presentation and justification of the research design which addresses the research objectives.

### 3.2 Research approaches in Requirements Engineering

#### 3.2.1 The Information Systems and Software Engineering disciplines

There is a tendency within the Information Systems community to perceive that there is a gap between the two disciplines of Information Systems and Software Engineering. Software Engineering (technology-oriented) is viewed as lying outside the Information Systems (business-oriented) research domain (Parker et al., 1994; Wafula, 1995). Traditionally, the Information Systems community has emphasised the human and organisational issues at the expense of the technical issues, which are an important and unavoidable part of Information Systems. On the other hand, “software applications are (and should be) embedded in a world of organisations, people and usage.” (Jarke
et al., 1993, page ). There is, in fact, overlap between Information Systems and Software Engineering; both these disciplines must be seen as equally important (Swatman and Swatman, 1992).

Paradoxically, requirements engineering is an activity which is both social and technical in nature. Indeed, requirements engineering is defined traditionally as an early phase of the entire life cycle of the development of both information systems and software applications. It concerns understanding human wants and their meanings in an actual organisational and business environment, then representing these issues in a manner which allows a technologist to develop a solution to the perceived problems. Clearly, it should be seen as a socio-technological activity. Requirements Engineering, thus, may be viewed as lying within the intersection of the Information Systems and Software Engineering disciplines. The research approaches considered appropriate to conducting research in Information Systems could also be applied for conducting research in Requirements Engineering.

### 3.2.2 Research paradigms in Information Systems: Positivism and Interpretivism

Information systems is a young discipline. Its origins and fundamental concerns are the effective use of information technology (computer systems) within organisations. Its epistemology draws heavily on the social sciences. Thus, the research approaches in Information Systems have generally been adopted from social science (Hirschheim, 1985).

There is a wide ranges of research approaches in the field of Information Systems (Galliers, 1991; Galliers, 1992; Shanks et al., 1993; Nunamaker and Chen, 1990; Avison,
Some common approaches are:

- laboratory experiments
- field experiments
- survey,
- systems development
- single case study, multiple case study, longitudinal case study
- action research
- grounded theory
- forecasting and future research
- phenomenological/hermeneutic studies

Galliers (1992) partitions research approaches into positivist (scientific or empirical) paradigm and interpretive (subjective) paradigm. Each approach represents a different way of looking at the world and is linked with a number of different techniques for data collection and interpretation. Choosing an approach involves choosing a way to observe and measure the world, even if some approaches may study the same thing or appear to produce the same results (Neuman, 1994, page 57). Rather than discuss each approach in detail, this chapter will outline the two philosophical viewpoints inherent in the two paradigms mentioned above, to form the basis for discussion and justification for the choice of the action research approach adopted for the project.
3.2.2.1 The positivist paradigm

This subsection reviews positivism in terms of its history, philosophical assumptions, basic principles, strengths and weaknesses, and current debate on the use of positivism in social research.

The origins

The positivist paradigm is favoured by many researchers from both applied sciences, such as physics, chemistry, and biology, and social science. This paradigm has also dominated the IS research during the 1980s (Mumford et al., 1985). Positivist social science is based on the work of the nineteenth century philosopher, Auguste Comte (1798-1857), the father of sociology. Many of its philosophical assumptions and principles are outlined in his major work *Course de Philosophie Positive* (1830-1842). The positivist philosophy has become widely accepted by social researchers many of whom insist that it is the only valid paradigm to improving human knowledge (Bleicher, 1982).

Positivist philosophy first originated from mathematics, physics, astronomy, chemistry, and biology and continuously developed through the history of human knowledge acquisition for centuries from the ‘dark age’ of western science (200-1000 AD) till today. It was extended and applied in social science in the early nineteenth century. Comte (1975) contended that all sciences “are not radically separate, but all branches from the same trunk” (Comte, 1975, p. 77) and “share a uniform manner of reasoning that is applicable to all the subjects matter that human spirit can occupy itself with” (Comte, 1975, p. xlviii). He insisted that natural laws govern all phenomena — both social and natural. Writing about the results of positivist philosophy he said: it was “the manifestation by experiment of the laws that rule the intellect in the investigation of truth and, as
a consequence, the knowledge of the general rules for that object” (Comte, 1975, p. 81). Implicit in this is an assumption that there is an objective world with its governing laws waiting “to be systematically and rationally investigated through empirical investigation” (Shanks et al., 1993).

**Philosophical assumptions and principles**

The positivist paradigm is based on an ontology in which the world is seen as objective and as existing independently of human perception. Its associated epistemological assumptions predicate that knowledge is hard, real and can be transmitted in tangible form (Burrell and Morgan, 1979) and the only way to acquire knowledge is to use hypothetico-deductive logics (theory analysis) and empirical data (observation), duly combined:

“Positivism defines social science as an organised method combining deductive logic with precise empirical observations of individual behaviour in order to discover and confirm a set of probabilistic causal laws that can be used to predict general pattern of human activity.”(Neuman, 1994, p.58)

Positivist researchers assume and argue that “there should be a relation of independence between researcher and his object. Inquiry is value-free and there should be a sharp demarcation between empirical observation and theory statements. There exists stable, uni-directional cause-effect relations in the world”(Jönsson, 1991, page 378)

These assumptions also provide the rigour and norms for the evaluation of the validity of research outcomes. It is very important that research outcomes can be reproduced and verified by other independent researches. Positivist research is ideally context-free and thus the knowledge discovered can be generalised.
In Comte’s (1975) view, the major components of the positivist method are developed by each science respectively, e.g. mathematics offers the elementary conditions of positivity, physics offers theory of experimentation, astronomy determines the study of nature through the scientific use (confirm and invalidate) of hypotheses, chemistry offers the art of nomenclature, and physiology offers the true theory of classification. According to Comte, mathematics, particularly its abstract part, is the cornerstone of the positivist method. Even though Comte cautioned its limited capability to deal with more complex problem like social systems, nonetheless, the limits are lying in our intelligence, he wrote, not in mathematics itself.

When applying these principles to social science, positivist researchers seek “to explain and predict what happens in the social world by searching for regularities and causal relationships between its constituent elements.” Burrell and Morgan (1979, page 5). Laboratory/field experiment, simulation, survey, forecasting future research, theory proof, and case study are approaches widely used by positivist researchers (Galliers1992). The researchers design measuring instruments, measure social phenomenon empirically, and test hypotheses by analysing data (which are often quantitative rather than qualitative) collected from the measure for the consistency (Neuman, 1994; Jönsson, 1991). The researcher is an outside observer and does not participate in the social phenomenon under study.

**Characteristics and their limitations in Information Systems**

Based on natural sciences, the positivist method is characterised by repeatability, reductionism and refutability (Checkland, 1981). However, Galliers (1985) argues all these characteristics are impossible when applying positivist approaches in Information Systems. The assumption that experiments can be replicated (Comte, 1975; Check-
land, 1981; Neuman, 1994) is not feasible for social phenomena, which are characterised by their uniqueness. Galliers (1985) repeats the saying “we cannot step into the same river twice” in this context. Antill (1985) also argues that no particular information system experiment can be repeated as the individuals, problems and situation are necessarily changed by the experiment. The second characteristic, reductionism, assumes that the phenomenon under study can be divisible into small manageable components. Unfortunately, each component, when being viewed apart from the whole phenomenon, might not be the same as in the interaction with other components (for more details see (Checkland, 1981; Galliers, 1985; Bleicher, 1980). The refutability characteristic concerns verifiable predictions. However, in contrast to non-human species, human species are conscious and have ability to learn and plan out activities: “completely autonomous and free-willed” rather than being controlled by the environment (Burrell and Morgan, 1979, page 6). This makes social life very fragile and influenced by the researcher’s predictions. Checkland (1981, p. 70) argues, “physical systems cannot react to predictions made about them; social systems can.”

Positivism holds the dominant role and is considered as scientific, perhaps due to the impartial and outside-observer role of the researcher in relation to the subject under study. Klein and Lyttinen (1985) found the strengths of positivism in its objectivity, rigour, and respect for data. Objectivity leads to high degree of reliability and validity (Kirk and Miller, 1986). The development of formal approaches and computer programming languages are examples of the positivist research paradigm in reducing the subjectivity of socio-organisational elements in information systems. On the other hand, they state that these are also the limitations of the paradigm in IS. Indeed, social action can only be understood by the actor(s) and can be translated to other people
through a subjective explanation of the researcher/observer. We cannot achieve absolute objectivity in a strict sense because this explanation of social action is strongly influenced by the researcher’s *weltanschauung* (Checkland, 1981; Galliers, 1985, p 215-221). The rigour and respect for the facts are also limited because social facts are open for different interpretations and conclusions reached from a series of same facts by different observers are often different.

In conducting research in different sciences, the positivist assumptions remain the same, however, their application may vary for different classes of phenomena. In studying human phenomena, (Comte, 1975) acknowledged that the social problems lay in society itself and in the complex structure of society. While criticising social science which at that time was in a stage of mere theological and metaphysical states and thus lacking in empirical rigour, (Comte, 1975) advocated that the positivist method adopted from natural science should be extended to study social phenomena (Comte, 1975). An alternative approach to social investigation can be found in the work of Wilhelm Dilthey (1833-1911), Max Weber (1864-1920), Edmund Husserl (1859-1938) and others authors in the late of nineteenth and through the twentieth centuries. During the last two decades, many Information Systems authors also question the use of positivist approaches in the discipline (Susman and Evered, 1978; Wood-Harper, 1985; Antill, 1985; Klein and Lyytinen, 1985; Nissen, 1985). They adopt the view that in contrast to physical systems, social systems are far more complex due to the consciousness of human behaviours and thus can not be studied successfully with the approaches from natural science (Burrell and Morgan, 1979). They criticise positivism as disregarding the context of the study (Susman and Evered, 1978; Jönsson, 1991; Neuman, 1994; Pettigrew, 1985). While natural phenomena can be isolated and studied, the social event shall be understood in its
setting. Interpretivism, an alternative paradigm for conducting research, is the response of social researchers, generally, as well as Information Systems and Requirements Engineering researchers (Galliers, 1992; Checkland, 1981; Wood-Harper, 1985; Bickerton and Siddiqi, 1993), particularly, to theses challenges.

**3.2.2.2 The interpretive paradigm**

This subsection discusses the interpretive paradigm in terms of its history, its underlying philosophical assumptions and strengths and weaknesses.

**Origins**

> “If one contends that social sciences embrace an epistemology which is different from their natural science counterparts, then so is the case for IS”

*(Hirschheim, 1985, page 13)*

The origins and needs of the interpretive paradigm lie in the dissatisfaction with the traditional positivist paradigm when studying unique human and complex social systems. Indeed, there is a growing recognition and use of the interpretive paradigm within Information Systems research (Mumford et al., 1985; Nissen et al., 1991). Soft Systems Methodology by Checkland and Scholes (1990), Multi-View by Avison and Wood-Harper (1991), investigation of organisational changes in system development by Orlikowski (1993), ethnography (Van Maanen, 1989), and FOOM (Fowler, 1996) are good examples among a number of applications of the interpretive approaches within the Information Systems discipline. Since this paradigm is relatively young, the philosophical assumptions and principles, the research’s validity, rigour and the researcher’s responsibility are still evolving in methodological debate.
Positivism has started to lose its dominant position with the emergence of interpretive social science. The origin of the interpretive paradigm can be traced back to the work of German philosophers, Immanuel Kant (1729-1804), Dilthey, Weber and Husserl. The paradigm then has been developed by other authors during the twentieth century. Based on the thought that knowledge can only be experienced and interpreted by the actor(s) the paradigm attempts “to understand and explain the social world primarily from the point of view of the actors directly involved in the social process” (Burrell and Morgan, 1979, page ). The paradigm emphasises the subjective aspects of the social world and the consciousness of human beings. Today, the paradigm is associated with various research approaches, such as case study (Eisenhardt, 1989; Yin, 1994), field study (Curtis et al., 1988), action research (Susman and Evered, 1978; Wood-Harper, 1985; Baskerville and Wood-Harper, 1996), ethnography (Van Maanen, 1988; Van Maanen, 1989), phenomenology (Boland, 1985; Rathswohl, 1995), and grounded theory (Strauss and Corbin, 1990; Strauss and Corbin, 1994).

**Philosophical assumptions and basic principles**

Interpretive philosophy could be described as based on the notions of *verstehen* (empathic understanding), *weltanschauung* (world view) and hermeneutics. The notion of *verstehen* can be traced back to the work of Dilthey and Weber, which emphasises socially meaningful and intentional action, which is argued to be understood as an object of a subjective interpretation and concerns the internal process of human minds (Burrell and Morgan, 1979; Hirschheim, 1985; Neuman, 1994). *Weltanschauung* in Dilthey is a totality consisting of people’s cognitive representation of the world, evaluation of life and ideas on how to conduct life. Hermeneutics (Palmer, 1969) is the study of interpretation of text and action. According to Schleiermacher and Dilthey, people carry their
subjective intention into the text and action (Palmer, 1969). The hermeneutic circle can be explained as the meaning of a whole obtained from each its parts in isolation being determined by the context of the whole. In other words, human activity cannot be studied in isolation from its immediate social context and the actors’ subjective *weltanschauung*. In contrast to positivism, the interpretive paradigm emphasises the intention and intra-action of social action, the context of study, the subjectivity of researchers and especially the importance of the social meaning.

The interpretive paradigm, thus, is based on an ontology in which the social world is seen as a product of consciousness, as subjectively experienced and open for different interpretations, and is subject to change on the basis of intention. Its epistemological assumptions predicate that knowledge is subjective, transcendental and can only be directly experienced and explained using the inductive analysis of qualitative observation data.

Interpretivism can be described as

> “the systematic analysis of socially meaningful action through the direct detailed observation of people in natural settings in order to arrive at understandings and interpretations of how people create and maintain their social worlds.” (Neuman, 1994, page 62)

**Characteristics, strengths and weaknesses of the interpretive paradigm**

Some important characteristics of the interpretive paradigm are naturalism, unique situation orientation, inductionism, and multi-faceted causal relationships (Patton, 1990; Jönsson, 1991; Neuman, 1994). In order to study and understand the social world, the research setting is a natural event or phenomenon where the researcher’s ability to con-
trol variables is minimised as opposed to controlled variables in conducting positivist research. The objectives of the researchers are the emergence of concepts, generation of insights and rich understanding of the unique special social phenomenon. In contrast to deductive analysis used by the positivist researchers, inductive strategies used by interpretive researchers focus on the emergence of meaningful explanation and common patterns from collected data, which are often qualitative rather than quantitative. In conducting inductive analysis, researchers take a holistic view and study the whole phenomenon rather than a small number of predetermined variables and linear cause-effect relationships. The research’s outcomes are interpretation and description of the phenomenon by the researcher rather than the discovery of objective causal laws. Although the written story given by interpretive researchers is not an objective representation of realities, it reveals experience and explanation by the researcher as an inner observer, immersed in the situation in order to understand, integrate concepts or revive and evaluate existing theories and conceptual frameworks. The inductive strategy is very fruitful for the explorative and discovery type of research rather than for testing hypotheses.

Thus, the strength of the interpretive paradigm lies in well-grounded data, which are often qualitative, in-depth descriptions of the reality including the context and fruitful and meaningful explanations of the phenomenon under study (Miles and Huberman, 1994; Eisenhardt, 1989; Yin, 1989). Qualitative data, which is often in the form of words and stories, are far more meaningful and convincing than numbers (Van Maanen, 1989; Miles and Huberman, 1994). The interpretive paradigm is preferred for its advantages for the creation of new ideas and concepts and for the understanding the process of the phenomenon under study (Jönsson, 1991; Galliers, 1991; Maxwell, 1996).
The weakness of the paradigm is that the findings are not value-free, are very much context dependent and have the unavoidable and unconscious bias of the researcher (Patton, 1990; Miles and Huberman, 1994; Galliers, 1991; Shanks et al., 1993). Interpretive research often requires a long period of time, a labour-intensive data collection process, and a large amount of unstructured qualitative data (Yin, 1994; Miles and Huberman, 1994; Eisenhardt, 1989). The uniqueness of the event or case under study and the subjectivity lead to difficulty in generalisation of the findings (Yin, 1994; Galliers, 1991; Miles and Huberman, 1994). Due to the qualitative data collection and subjective and not well-formulated inductive analysis method, the reliability and validity of interpretive research have been a great concern and have been addressed in the literature (Dick, 1993; Miles and Huberman, 1994; Kirk and Miller, 1986; Maxwell, 1996; Mason, 1996; Neuman, 1994).

The objectivity vs. subjectivity discussion is an ideological debate in the research methodology literature, for example Neuman (1994) and Patton (1990). An ideal objectivity cannot be achieved when studying human beings and social systems while, on the other hand, subjectivity limits the credibility and generalisability of the research. Each research paradigm has its strengths and weaknesses. In recent years, there has been a growing acceptance of the pluralism of research paradigms and approaches in the Information Systems community and a discussion of strategies for choosing an appropriate approach for a particular research situation according to a specific stage of knowledge (Galliers, 1992; Fitzgerald, 1991; Shanks et al., 1993; Fowler and Swatman, 1996). This will be discussed in more detail in the following subsection.
3.2.3 Choosing an appropriate research approach for the project

In the last few years, there appears to have been an assertion by many methodologists that rather than there being only one correct approach to improve human knowledge, there are a number of different approaches considered to be appropriate in specific research circumstances. Fitzgerald (1991) propounds recognition of strengths and weaknesses of different research approaches and the possibility of the use of multiple approaches.

Galliers (1992) analyses the advantages and disadvantages of different research approaches in Information Systems and has proposed a taxonomy, which is a guide to which research approach to employ in which circumstance. In his paper, “the perspective of the results/conclusion” (Galliers, 1991, page 341) of research can be used to select “the most likely to be appropriate approach for one’s study” (Galliers, 1991, page 340) . Briefly, research approaches are considered in the context of theory building, extending, and testing theory. Shanks et al. (1993) add some further issues for researchers to consider when identifying an appropriate approach to their research. These factors are audience, framework, purpose (exploration, descriptive, or explanation (Neuman, 1994)), context and also stages in the research cycle.

When discussing research in the domain of Requirements Engineering, Fowler and Swatman (1996) recognise and strongly confirm that the choice of research approach should be made by taking into consideration the suitability of both the particular research circumstance and also of the stage of knowledge of the phenomena.

Having carefully studied these issues, especially, the objectives of the research, the stage of theory and the research circumstance, I decided to consider case study and action
research as possible research approaches to be used. The following two subsections will discuss case study and action research to provide a basis for the justification of the choice of action research in section 3.3.2.

3.2.4 Possible research approaches

An interpretive approach is sought for the project for a number of reasons:

- the exploratory character of the research questions and the immaturity of concepts,

- the objective of the development of FOOM to be practical in order to improve current practice,

- the youth of FOOM,

- the unfamiliarity and the current biases towards the use of design rationale by user participants (Shanks and Darke, 1997). According to Shanks and Darke’s (1997) study, design rationale is time consuming and expensive to document and is not useful for user participants.

The first two reasons state the objectives of the research and the last two reasons state the current stage of theory and the circumstance of the research. The first reason does not allow a small number of variables to be observed and measured. The second reason drives me towards a real case for requirements specification in the development of a software application. The subject under study is a complex socio-technical process of applying FOOM for the investigation, including the formation and evaluation of new ideas. The data, which tend to be a descriptive and meaningful explanation of the process, will be more likely qualitative. The research, thus, requires a holistic perspective
and deep understanding by the researcher to generate insights and to investigate and explore new concepts. The research outcomes will be new suggestions to improve current FOOM. According to the two last reasons, the current stage of theory and research circumstance are not yet ready for observation merely from outside, the research, thus, requires an active involvement by the researcher.

Due to the importance of the involvement of the researcher, naturalistic preference, and the limited time permitted for the project, case study and action research are chosen for consideration.

**Case study**

Case study involves an in-depth examination of a particular issue in an organisation/event environment (Yin, 1994). Though including case study in the positivist stream of research, Galliers (1991) acknowledges that one could also consider case study as an interpretivist research approach. Neuman (1994) argues that case study may be seen as interpretive rather than positivist since there is the possibility of different interpretations of observed facts by the researcher.

Instead of gathering specific information on a large number of cases to analyse quantitatively, researchers may study intensively a small number of particular cases. There are various forms of case study research, such as single case study, which involves only one particular organisation; multiple case studies, which are done in some organisations to compare and strengthen findings across cases (Yin, 1994; Eisenhardt, 1989), or longitudinal studies to better distinguish cause and effect (Pettigrew, 1989).

Case study, thus, is very much contextual rather than context-free, as is positivism.
The strength of case study research lies on its ability to provide a more detailed and richer description of a social problem in a real setting rather than the one gathered by capturing a snapshot of reality by using survey research.

However, as far as the research is context dependent, it is also characterised by the uniqueness of the context and thus limits the generalisation of the outcome. In addition, the influence of the researcher on observed people and on organisational events is impossible to eliminate. Also, the researcher’s subjective explanation of facts and biases is not deniable. Case study, thus, requires high responsibility from the researcher to check data collected (Neuman, 1994). The outside observer role to the subject of study of the researcher often makes the control over variables difficult (Galliers, 1991; Shanks et al., 1993).

**Action research**

The action research approach is similar to the case study approach in that it involves detailed study of a specific problem within a single or group of events/organisations. However, action research differs from case studies in that an action research study acknowledges the active and reflective involvement of the researcher in a purposeful action (Galliers, 1992; Shanks et al., 1993):

> “the researcher is within the field of that research and becomes a partner in the action and process of change” (Wood-Harper, 1985, page 178).

Action research is a conscious effort of the researcher to apply a theory in a real-world situation to test the theory and in turn, provide practical outcomes for theory building (Galliers, 1992; Wood-Harper, 1985; Wood-Harper, 1992). This is where the researcher
makes a contribution to theoretical knowledge.

Action research may be classified within the interpretive research category (Wood-Harper, 1992; Galliers, 1992; Jönsson, 1991; Avison, 1991; Shanks et al., 1993). Obviously, action research is not positivistic research. Positivistic research holds to be context-free and produces results which can be generalised, while action research/case study is contextual and the results are rarely generalisable in a strict sense, because each event/organisation is unique (Galliers, 1992; Wood-Harper, 1992). Moreover, in the former, researchers set their aims and play as observers while in the latter researchers negotiate aims and roles with clients and act from the position of membership (Checkland, 1991; Wood-Harper, 1992). The approaches used in positivism and action research are clearly different. In the former hypothetico-deductive logic can be applied while in the latter the outcomes are qualitative rather than quantitative, and will depend on the researcher’s interpretation (Jönsson, 1991). Jönsson (1991) adds that in action research learning takes place during the course of action while in positivistic research learning “takes the form of theory-based conjectures” (Jönsson, 1991, page 390). In Checkland’s view there must be an explicit methodological framework “declared in advance, in terms of which learning will be defined” (Checkland, 1991, page 397).

The action research process is described differently by many authors — in fact it would be more accurate to say that the literature describes a spectrum of processes which belong to the action research “family”. Susman and Evered (1978) define the action research process as including a number of phases: those of diagnosing the problem, planning, taking action, evaluating the effect of the theory, and specifying the findings. Checkland (1991) describes a cyclic action research process which consists of the selection of a real-world situation (A), establishment of roles of participants, declaration of
an intellectual framework (F) and a methodology to be used and tested (M), reflective
activities of the researcher to adapt the F and M, rethinking of the three last stages,
and review of experience and learning concerning the A, F, and M when exiting out
of the situation. Overall, the researcher enters a real situation, places him/herself in
the context, and generates qualitative rather than quantitative data as outputs of the
project; thus the intervention of the researcher influences the outcomes of the research
(Galliers, 1992; Jönsson, 1991).

The emancipatory outcomes, practical and theoretical benefits of the research process
are advantages of the approach (Galliers, 1991; Galliers, 1992; Shanks et al., 1993).
Similarly to case study, the limitations of the approach are the restriction to a single
event/organisation, the difficulty in generalisation of the results, the subjectivity of the
approach, the inability of researchers to be unbiased, and the different explanations of
researchers of events (Avison, 1991; Galliers, 1991; Galliers, 1992; Shanks et al., 1993).

Both case study and action research are seen as suitable for exploratory research to
generate issues naturalistically and inductively (Shanks et al., 1993; Eisenhardt, 1989;
tions with little else and a wish to investigate them in more or less natural setting, I
first thought case study approach as an option. However, it was difficult to find an
organisation willing to apply a new method and ideas of capturing design explanation
to their commercial project. Having carefully considered the issues related to finding a
host organisation and the issue of the ability to control research process to pursue my
objectives I decided to select action research. More details about justification of choice
will be discussed in the section 3.3.2
3.3 Justification of research approach and research design

3.3.1 A programme to develop FOOM — the research context

FOOM was proposed by Swatman between 1990 and 1992 and since then has evolved in a research programme which conforms to an adapted cycle of theory evolution. In the first stage, an argumentative research project has been undertaken to review the poor development practices in Information Systems and Software Development and to derive a solution from theory (Swatman, 1992a). The second stage is of preliminary validation of the arrived method. A number of pseudo-laboratory experiments, simulated trials, small field trials and training courses have also been conducted in order to test the feasibility and feed back to the new theory in the university research environment (Swatman et al., 1991; Swatman, 1992b). After this stage, the theory reached a satisfactory level of theory soundness. The third stage involves the evaluation and improvement of the method based on the reflection upon the application of the method in real settings.

The outcome of action research by Fowler (1996) has further enhanced the method and contributes to it’s both theoretical soundness and practical aplicability. Beside this, the general findings and specific insights into complex modelling process point out the need for the further evaluation and development of the theory. My project, thus, could be seen as a step in the extension and a further testing of the method in the requirements specification of a real software application project. The research objectives involve an intensive exploration and should acknowledge a subjective understanding (explanation) of the requirements engineering process in a specific context. This means the research outcome would involve a responsive interpretation of actual requirements
engineering process and activities from the requirements engineer point of view to the reader point of view. New possible concepts are to be based upon this interpretation. I decided to trade-off the deductionism and controlled experiments of traditional scientific approaches for the inductionism and holistic-naturalism of the interpretive approaches. I choose to record the process/facts to be interpreted in the form of qualitative data and not to restrict observation and measurement to only certain variables. Moreover, as analysed in the previous section an active role of the researcher (not merely an external observer) is required. The frequent move between actor and observer during the action and the interpretative observation suggests to me that I adopt action research as the most suitable approach for the project. The following subsection will give a detailed explanation of the choice.

3.3.2 Justification of action research as a valid choice

Wood-Harper wrote:

“if real insight is to be gained from real life situations...there is no other alternative than to use some form of Action Research.” (Wood-Harper, 1985, page 178)

This project involves the application of a new information systems specification/development method in order to answer two research questions. A secondary issue may be the feasibility of the method. The research is classified as investigatory; the evolution of requirements during the process of applying FOOM, and the use of design explanations to enable and improve the traceability of the development process of requirements specification will be studied in a real situation. This addresses
the key principles of action research approach described by Galliers (1991, 1992) and
Shanks et al. (1993). This also addresses what Jönsson (1991) states as a strength of
interpretivism in the emergence of concepts and theory from the study of actions.

Particularly, in this project the FOOM approach was used to design a FOOM CASE
tool, which enables the recording and management of design rationale in the form of de-
cisions and relating arguments explaining and justifying the evolution of requirements.
At the same time, the CASE tool supports the generation of different argumentation
notations used to construct explanation documents. This will result in a specification
in Object-Z and in modified MOSES diagram linked via a semantic mapping developed
by Wafula (Wafula, 1995; Wafula and SWATMAN, 1995a; Wafula and SWATMAN, 1995b).

Questions and issues concerning requirements changes, decision making and the use
of argumentation notations, particularly, and the practicability of FOOM, generally,
which appear during this process will be recorded. The data gathered will be analysed
to answer two research questions.

In designing the research project, I encountered the dilemma that the project could be
undertaken in an organisation/commercial environment, which better reflects reality, or
within a university research and development project (non-commercial environment).
There are some issues which have been thoroughly studied.

• It is hard to find organisations willing to apply a very new method in a real
business project. Moreover, FOOM, is not yet complete. It would not therefore
be appropriate for a real commercial project at this stage of its development.

• FOOM, with its mathematical concepts and notation, requires a training course
for the participating Information Systems staff of the organisation and for the
user validator(s).

- Although practitioners would benefit from the system (when developed), it is very hard to persuade outside organisations to be a host for a project when the core concepts are still at the early forming stage.

- Given the strict requirements of the system to be developed in order to support research interests, it would be hard to negotiate with outside participants. In other words, when pursuing research objectives I would prefer to have a special kind of control: to support a relatively flexible process (of the development of the application) in order to investigate the new concepts (initial and generated during the process), and to reflect reality as much as possible.

Having considered all these reasons, action research conducted in a development project in an university research environment, is most suitable approach. The researcher herself applies design explanation within FOOM while, at the same time, observes her and other participants’ actions to investigate and explore the benefits.

3.3.3 Research design

**Research setting**

The project was undertaken in the Centre for Information Systems Research, a university research and development environment. A small team of three, a FOOM specifier/researcher and two participants, a FOOM developer and an expert in design explanation, were involved. During the process of development, a group of academics and Ph.D. students in Requirements Engineering, the Object-Oriented approach and design explanation were also involved at some stages. The overall requirements engineering
The process of an action research project can be described as the process of purposeful, in-depth and intensive learning from the researcher’s experience (in the action) and his/her understanding in order to propose possible changes and improvements to a methodology being evaluated. In this project a cyclic approach, as described in Checkland (1991), is adopted and modified. The adapted action research process consists of

1. entrance into a real-world situation (A) and establishment of roles of participants,
2. planning foci for action and observation and declaration of an intellectual framework (F), a methodology to be used and tested (M) and application (A) where the methodology is used,
3. actions performing changes and observation of the process,
4. interpretation and analysis on observation data, reflection on F, M and A,
5. either go back to step 2 or exit out of the situation and reflection on experience and learning concerning the A, F, and M.

Figure 3.1 describes the action research process used in the project. The process consists of ongoing research research cycles. Each research cycle consists of steps 2, 3 and 4. F, M and A are intellectual framework, methodology to be used and real-world application respectively. F”, M” and A” denote any temporal stages of these framework, methodology and application resulting from each research cycle and used in the
subsequent cycle. $F'$, $M'$ and $A'$ are outcomes of the study. In addition to Checkland’s (1991) description of action research process, the adapted process explicitly acknowledges changes to the application $A$. I believe when declaring and rethinking $F$ and $M$ in each cycle we should also pay attention to the intermediate state of the product. In fact, together with the process, the product could become a subject of change. This happens in this project.

The action research cycle can be characterised by hermeneutics as described in Palmer (1969), Carr and Kemmis (1986), Susman and Evered (1978), and Dick (1993). Hermeneutics is the interpretation or explanation of experience (Rathswohl, 1995). Each cycle involves intended action, data collection, learning and refining understanding. Carr and Kemmis call this “interpretive understanding” of the problem. This, in turn, leads to a new course of action for the next cycle. The preunderstanding of each cycle is based on the understanding of the previous cycle (Gummesson, 1991).
There are two important issues that should be considered when designing research structure.

- **When to stop a cycle?**
  
  A cycle is often started with some assumptions and involves data collection and a preliminary analysis. The critical moment is when the collected data are no longer insightful to address the initial assumptions and when questions generated from data collection/analysis suggest a significant change in the course of action and/or initial assumptions. We believe this is the time to stop the collection of data for the cycle and to evaluate what has been observed. The answer to the initial questions, and newly generated questions should form a structure for the next cycle in terms of (new) $F$, $M$ and $A$.

- **When to stop the study?** (not to go for further cycles)
  
  We can stop the study when newly generated questions are not insightful and, most importantly, when they do not suggest any significant change to the course of action and the research assumptions.

**Two research cycles of the project**

The project is undertaken in two action research cycles. Rather than being predetermined before the start, the two cycles are driven by the learning during the course of action. This fact is consistent with what is described in Jönsson (1991). Here is a short description of these research cycles in terms of $F$, $M$ and $A$ described in the previous subsection.
1. **Introducing process-oriented design explanation within the FOOM process**

During the first cycle, IBIS, an *ad hoc* notation of design explanation is introduced to record and explain the modelling process including the evolution of requirements, modelling decisions and their rationale. **F** is the framework of ideas described in the research objectives, **M** is using IBIS within FOOM and **A** is the development of a Meta-Argumentation Workbench (MAW). More details are provided in Chapter 4.

The evaluation on the observation of the use of IBIS indicates both its usefulness in recording the graph of the evolution of requirements versions and its limitation in evaluating versions or different possible requirements modelling solutions. The limitation of IBIS suggests the need of a *post hoc* notation for the evaluation of requirements modelling solutions. The FOOM structure diagrams show their usefulness and effectiveness in requirements representation and communication between specifiers and between specifiers and user participants. The evaluation of the application (**A**—MAW) leads to an extension of the **A**. **A** then is considered to become a FOOM super-structure management CASE tool in order to support recording and structuring the evolution of requirements and the underlying explanations. This suggestion leads to the need and new foci for the second research cycle.

Interestingly, IBIS happens to be very useful in recording and structuring qualitative data in this action research. This leads to a new question: Can design explanation notations (IBIS and QOC) be used to structure qualitative data in action research? This side issue requires attention in further research cycles.
Details of the first cycle will be described in the following chapter.

2. Adding structure-oriented design explanation

The second cycle focuses on adding the use of QOC, a *post hoc* design explanation notation to overcome the limitations of IBIS. The observation focuses on the potential benefits of supplementing IBIS with QOC as well as the similarities and differences between the two notations when using them within requirements engineering. The focus on the evaluation of the use of both IBIS and QOC within the modelling process remains one of the main research questions. The learning is be defined within the framework (F) which is derived from the discussion in Chapter 2 and is strengthened and further refined with the finding from the first research cycle. M now is using both IBIS and QOC within FOOM and A becomes a FOOM super structure management CASE tool.

The requirements become far more complex than expected. The use of FOOM, especially the formal Object-Z and the dynamic diagrams, is very desirable in better understanding and revealing conflicts in requirements and in representing and validating requirements with user-participants. The evaluation of the observation data suggests a systematic approach to using both design explanation approaches, IBIS and QOC, within requirements engineering.

The findings confirm the usefulness of the systematic use of the two design explanation strategies within FOOM and suggest further research into its validation in different situations.

Details will be described in Chapter 5.

Therefore, this thesis involves an application of and reflections upon F, M and A, described above. As a result, the study also leads to an deep understanding of the
opportunistic nature of the process, and examines the oscillations in complexity using IBIS and QOC. The thesis extends the framework and suggests that design rationale could assist project managers in understanding the RE process and its dynamics (F'). The study also leads to a new approach (M') to using design explanation within FOOM. Details will be provided in Chapters 6 and 7.

The application results (A') is a requirements specification of a FOOM CASE tools. According to this, the initial workbench has become a tool to manage the evolution of a FOOM specification and record the design explanation relating to this evolution. Design explanation can be structured using either IBIS or QOC notations.

Qualitative data collection and analysis

Existing options

Qualitative data collection and analysis play a vital role in conducting qualitative research in general and action research in particular. Qualitative data serve as empirical evidence and explanation for the research outcome. Various techniques have been proposed to collect, represent and analyse the qualitative data which often appear in the form of words rather than numbers.


Furthermore, Van Maanen, Patton, and Neuman are representative of a section of the academic community which advocates the narrative prose representation of qualitative
data. The advantages of this approach lie in the richness and completeness of the data. It is especially helpful when the researcher does not know which data may be useful in the future. However, the approach leads to a poorly structured body of data, which is very hard for researchers to analyse. Moreover, the volume of data may be daunting due to the open-endness of the research problem (Eisenhardt, 1989).

Van Maanen (1989) points out that each research project has its own difficulties arising from their different theoretic and methodological perspectives. A challenge central to this thesis is how to collect and analyse data for the study.

**Choosing an appropriate option**

The goal of data collection in this thesis is to generate an intimately focused and rich description and explanation of the process being evaluated (Patton, 1980). Therefore, I adopt Patton’s (1980) argument:

> The purpose of observational data is to describe the setting that was observed; the activities that took place in that setting; the people who participated in those activities;...Observational reports must include sufficient descriptive detail to allow one to know what has occurred and how it occurred. (Patton, 1980, p. 124)

In this action research, the research diary must focus on how and why a requirements specification is being produced rather than what is produced. The data describe the various participants’ perceptions of the process of using design explanation within FOOM. These data are linked to similarly structured data which describe the agreed perception of the process of the evolution of the specification artifact, specifically, the construction of O/C and events chain diagrams and the mapping between them, and the commu-
communication between user and specifier to determine and validate the requirements. As a longitudinal study (over 18 months) was involved, the traditional textualisation process would produce masses of unstructured prose that would make effective data analysis vulnerable. On the other hand, since I began the process without formal criteria against which to explore and evaluate the use of design explanation within FOOM, I would prefer not to build tables/matrices, which may restrict the ability to generate questions and insight.

Therefore, the qualitative data to be collected should be semi-structured and should aim at describing the requirements engineering process and the experience of the researcher during the course of action. The data include both design documents and the design process (e.g. as people, notes, details of conversations and activities, difficulties faced and overcome). Data sources could be:

- observational data including transcripts of the design conversations, activities, and decisions structured using a design explanation notation
- notes taken from requirements analysis sessions
- notes taken from unstructured and semi-structured interviews with the participants
- technical documents such as records of intermediate versions of the specification including according O/C models, events chain diagrams, Object-Z specifications, and textual documentation

The methods used to collect observational data include document reading, notetaking, and discussion (informal/semi-formal interview) between the participant-specifier and the clients. The data are not constrained by any predetermined categories. In fact, the
researcher collected all kinds of data, on any aspects of the problem to represent the complete picture of the situation.

This information is gathered and recorded in a research diary for interpretation and evaluation. The researcher occasionally developed semi-structured ‘questionnaires’ in the form of a list of issues then applied them in post hoc interviews with her participants in order to examine/cross-examine their perceptions of the usefulness of design explanation.

The researcher’s qualitative analyses were validated through discussions with expert panels. In this research, at times, the results from these analyses were represented and discussed at research meetings of a requirements engineering research group at Swinburne University of Technology and Deakin University. These results were also published and discussed in a number of academic papers in journals and conferences in Australia and internationally addressing various audiences, including primarily academics and also practitioners, in Requirements Engineering, Information Systems, and Software Engineering (see List of Publications attached in the back of the thesis).

Feedback and comments received from communicating and discussing research findings with the audience increased the confidence of the findings, highlighted directions, and in fact, were incorporated in further observations and analyses.

Data collection and analysis for each of the two research cycle (see above) will be described in more detail in Chapters 4 and 5.

In conclusion, action research is chosen and justified as an appropriate approach to address the research objectives. The research design used in the project is described and explained.
Chapter 4

Using ad-hoc Design Explanation within the Requirements Engineering Process

In this chapter, I describe the first hermeneutic cycle of the research programme. A brief introduction to the research cycle has been provided in Chapter 3, Section 3.3.3. Specifically, the intellectual framework (F) adopted in the research cycle is derived from the discussion in Chapter 2: the systematic documentation of arguments made during the requirements engineering process may explain the evolution of requirements specifications and improve the understandability and traceability of the requirements specifications. The method (M) being explored is the incorporation of the IBIS approach within FOOM. The real world application (A) is the specification of requirements for a support tool (see Section 4.1.2). The research cycle is described in terms of research process and findings.
4.1 Research process: observation and data collection

This section explains why IBIS has been chosen from numerous design explanation notations, it outlines the requirements engineering project used as a real world application for the study, and it describes the process of using IBIS within the requirements engineering process.

4.1.1 Choosing a process-oriented design explanation notation

Having argued that the requirements engineering process cannot be deterministic (see Chapter 2), I have decided to take a descriptive approach to documenting requirements evolution and the rationale for the evolution. An appropriate notation to be used for the study should have the following properties:

**descriptiveness** The notation should have the potential of intimately describing the on-going process of requirements engineering.

**non-intrusiveness and simplicity** Extra effort for the training and documenting of the requirements engineering process should be minimised.

‘Descriptiveness’ suggests that an *ad hoc* approach (one in which the rationale for decisions is recorded as decisions are made) with the focus on process would be relevant. Various process-oriented notations considered for use in the study include IBIS and a number of its variations (itIBIS, gIBIS, PHI and REMAP), Toulmin, the Potts and Bruns model, COED and EOM. A description and discussion of these approaches is provided in Chapter 2.
Toulmin, COED and EOM were not chosen due to their complexity. These notations are based on a complex set of argumentation components and complex linkages between these components. Indeed, the Toulmin model requires a sophisticated classification of concepts (Backing, Warrant, Grounds, Qualifier, Claim and Rebuttal) and the judgement of a chosen Claim through a number of linkages of On-account-of, Since, So, Therefore, and Unless. The Toulmin model was designed with a specific focus on the justification of legal decisions and may not be flexible in respect of use in other domains. Especially, it is not suitable to describe the construction and documentation of initially vague and ambiguous IS requirements problems. Although COED is designed for naturally structuring the design conversation in a specific context, it also requires a complicated classification of utterances (commitment, directive, assertion, declaration, emotive) and representation of their characteristics (such as weight, volume, indicator, etc.). The EOM notation is rather complicated due to different levels of abstraction involved and its reliance on support tools. Moreover, EOM tends to capture the mutation of the design rather than to record the underlying rationale (see Chapter 2). Therefore, these notations would require extra effort in understanding, classifying the designer’s thought and translating them into these concepts. We have been unable to develop a rationale to suggest that any such effort would be beneficial in the requirements engineering process—in fact, we suggest they may obstruct the documentation of the process.

Potts and Bruns’s (1988) generic model may be simple, however, it needs to be refined and tailored to a specific requirements engineering method’s vocabulary before it can be used under study. This may constrain the researcher in generalising from the outcome of the research, especially when the generic model itself and its underlying concepts may not be well developed and evaluated sufficiently (see also Chapter 2).
IBIS has the required properties of descriptiveness, intuitiveness and simplicity,—and is also popular. IBIS and its variables are widely used—IBIS itself was tested and used in a number of large-scale, real-world projects in design and planning during the 1970-80s. In addition, empirical studies have found that itIBIS is intuitive and useful (see Chapter 2).

Major IBIS variations have been considered. Unfortunately, the reliance of REMAP and gIBIS on tools supports and the hierarchical structure of Issue in PHI (see Chapter 2) reduce flexibility and speed in note taking during requirements discussions. Moreover, PHI support tools were designed specifically for construction design, not requirements engineering.

itIBIS—a textual form of IBIS, has been chosen. Simple both to learn and to use, itIBIS is also flexible, it may be used irrespective of the technology available—not even a word-processor is required.

4.1.2 The requirements engineering project

The project under study was the requirements engineering phase in the development of a design support tool. The project was undertaken in a university research and development environment in Australia. This investigation took place over 6 months. It involved a small team consisting of: a FOOM specifier/researcher, a FOOM developer and an expert in design explanation. All these participants are requirements engineers. Both the FOOM developer and the design explanation expert worked in the industry for a long period of time as requirements engineers. The FOOM specifier/researcher worked in the industry as a programmer.
The project which forms the basis of study was inspired by the Meta-Argumentation Workbench (MAW), developed by Shanks et al. (1994). MAW was based on a meta model which was found to underlie a wide range of design explanation notations. Our requirements were to develop a customisable tool which supports various design explanation notations through remodelling MAW in an Object-Oriented style and enhancing it in a variety of ways including: supporting the integration of design explanation and design objects; and recording the dynamics of the documentation process. The project was, at this stage, called Notation Generic Argumentation Workbench (NGAW).

4.1.3 Using IBIS within the FOOM process

The FOOM process can be described as cyclic, each cycle consisting of elicitation, modelling and evaluation (validation) activities (see Figure 2.15 Chapter 2, for more details). During the modelling process, *ad hoc* arguments about requirements problems were structured using the IBIS notation every time a decision was made. Each IBIS argument represents a specific, local requirements problem/decision. When it came to evaluating different modelling options, relevant information was extracted from the IBIS base. IBIS arguments were also used during the elicitation activity.

The IBIS arguments were associated with the specification to explain decisions made. Therefore, the IBIS base grew in parallel with the evolution of a FOOM specification expressed in different forms of object-oriented diagrams and a formal Object-Z document. To support documentation of the process, requirements models were versioned. Figure 4.2 illustrates the process of documenting the evolution of both FOOM models and IBIS documents.

This process of applying IBIS within FOOM was observed and recorded in a research
diary. My primary activities included both the capturing and coding of requirements discussions and decisions using the IBIS notation as well as taking part in the requirements analysis and modelling process. Intermediate versions of the requirements model and changes to the versions were also documented. The IBIS documents were attached to the intermediate versions and associated changes.

Research data included:

- FOOM requirements discussions and modelling decisions structured in the IBIS notation. More than 350 Issues and Sub-Issues were recorded in the research diary. Each Issue/Sub-Issue is associated with a number of Positions and Arguments.

- Questions and issues concerning the use of IBIS within the requirements engineering process documented in the form of notes or Issues without Positions and Arguments. These include notes taken from semi-structured interviews between the researcher and the participants as well as the notes on thought processes taken by the specifier/researcher.
Figure 4.2: Documenting the FOOM requirements evolution and the rationale using IBIS arguments

- Intermediate states of the FOOM models (object-oriented diagrams, Event Chain diagrams and Object-Z diagrams).

The data were qualitative and supported my interpretation and analysis presented in the next section. The requirements modelling and also the observation and data collection activities required me to be flexible in switching between the roles of actor and interpreter.

4.2 Reflection on the use of IBIS

4.2.1 A summary of findings

The evidence, which is described in more detail in the next subsection, shows that

- the IBIS notation provides a useful and effective mechanism to describe the FOOM process, including discussions about requirements and modelling activities,
the IBIS notation also supports communication between participants\textsuperscript{1},

- the IBIS arguments provide a useful source for the extracting of relevant rationale information when needed.

but that the approach exhibits some limitations:

- the locality of IBIS arguments and

- the lack of context where an Issue is discussed.

Due to these limitations, extracting the IBIS information which documents each specific construct or decision becomes problematic.

4.2.2 An analysis of findings

During the process of acquiring and analysing requirements for the support tool (see Section 4.1.2) all conversations between specifier and clients were recorded and structured in the itIBIS form. The requirements engineering activities including discussions, reasoning and decisions about requirements and extensive changes to object-oriented diagrams and Object-Z documents were documented as they occurred. The IBIS notation with its narrative spirit was found to be useful for dynamic recording identification and refinement of requirements (through discussions about requirements). Ambiguities, hidden problems and assumptions were highlighted naturally as requirements discussions were structured in the form of Issue, Position and Argument. Let’s examine the following example closely.

\textsuperscript{1}We use participants to refer to all people involved in the delivery of a complete FOOM specification including specifier and clients.
One of the primary tasks of the requirements exercise was to support various argumentation notations as well as the creation of argumentation documents based on these notations. An argumentation document has been defined as a node-link diagram constructed according to a notation. A notation defines a set of node types, link types and linkages allowed between them. A question raised during a conversation between three participants is whether to allow involuted links in node-link Model/Argument\textsuperscript{2}, i.e. whether a link can connect a node to itself. The following Issue A\textsuperscript{3} records this conversation.

**Specific Issue A** Specifier: Is it possible to create a link connecting a component within an argument to itself within any existing design rationale notation?

**Context** To specify a constraint for Link in Object-Z. The specifier asks two other participants for clarification of the relationship between an argument’s components. In trying to answer the question, we broaden the context of argumentation to modelling notations in general.

-P 1 No, the involute link is not allowed in Arguments. (Suggested by The specifier and Participant 1.)

AS Specifier: For example, the marriage relationship between two persons, as in Entity Relationship modelling. However, in IBIS I cannot find any link from a specific Issue, Position, or Argument to itself at the level of instances. What about other notations?

AS Participant 1: (agrees and confirms that we cannot link an issue instance to itself.)

-P 2 Participant 2 suggests specialising the notation.

AS Participant 2: The approach we are approaching allows us to define rules on

\textsuperscript{2}Italic words such as Model and Argument denote objects/classes in the requirements model for NGAW. See notational conventions, Page xvi.

\textsuperscript{3}Issues were coded according to the date and time and the order of creation, however for the illustrative purpose in the thesis, I prefer to use capital letter codes
connections but only at the class or type level. There is no facility for us to define rules at the object-instance level (e.g. to disallow involuted link).

**Sub-Issue A1** How do we disallow involute links?

- **P A1.1** Specifier and Participant 2: Define a subclass of link called non-inv-link with constraint.

  **AO** Specifier: We choose not to do it for simplicity.

  **P 3** Specifier: The involuted link is allowed.

  **AS** Participant 2: Give users the flexibility to decide when to use involuted links.

The problem was discussed and understood. Two positions were considered. The second position suggested a further problem (**Sub-Issue A1**) to be solved. The above issue, together with its sub-issue, documents and explains the following change to an Object-Z specification:

The above example was chosen among many other issues because, for all its simplicity, it illustrates the usefulness of IBIS in describing the process of requirements under-
standing and modelling. Within the context of an Issue, IBIS information describes a conversation and the reasoning of the participants in solving a problem whereas a number of related Issues and Sub-Issues describe the path of understanding a problem and reaching their solutions.

Let’s examine a number of other IBIS Issues. Issues B, C and D also show the process of understanding and solving a problem in the requirements engineering project.

Issue B clarifies our understanding of the class Decision in the context of the relationship between Argument and Model. An Argument discusses a Model and may support changes to the Model. A Decision records the outcome of the Argument, i.e. to or not to make changes to the Model. This Issue leads to a Sub-Issue (the concepts of Issue, Sub-Issue and Context will be discussed more formally in the next subsection).

**Specific Issue B** What is a Decision?

**Context** To clarify an understanding of the relationship between three classes Decision, Model and Argument. This issue, therefore, is related to a number of arguments around this relationship (one ternary or two binary ones?).

**P 1** A Decision records references to the evolution of Model(s) AND the underlying rationale.

**AS** This definition supports the relationship between Argument and Model.

**AS** This definition expresses the relationship between the three classes elegantly.

**P 2** A Decision records a set of Arguments which support that decision.

**AO** We may have a Decision with an Argument not related to the relevant Models.

**AS** This definition supports the specialisation of the class Decision to two sub-classes. See the following sub-issue.

**Sub-Issue** Should we have two kinds of Decision?

**Context** There are two types of Decision: to change or not to change a Model.
We have already decided on an acyclic structure for the evolution graph, therefore there must be two Models involved in each Decision. These Models may be different or identical (two copies of the same content). This approach, however, does not reflect both the static and dynamic states of requirements clearly. Should we have two kinds of Decision?

?P Not to specialise Decision and record changes to a model using δ.

?P To specialise Decision to two subclasses: to support a transition from a Model to another different Model and to support not to change a Model.

The problem of representing the rationale and its outcome using the classes Decision, Argument and Model and their relationship was unsolved at this stage. Later, this problem was reconsidered in a number of other discussions and was solved. Issues C and D are two examples of such discussions. For brevity Arguments are excluded from the extract.

Specific Issue C How to understand the relationship of Decision, Argument and Model.

Context In the current model there are three binary relationships between Decision, Model and Argument. However, a decision for a transition involves two models and an argument. We discuss how best to represent this.

*P Keep the current model with three binary relationships between these classes.

...

Specific Issue D How to record static state, changes and dynamic transitions of Model.

Context We discuss recording decisions about requirements and how to attach arguments to models and transitions. (See the related issue on how to understand the relationship of Decision, Argument and Model, Issue C.)

*P This can be modelled as a ternary relationship between these three classes.
The above examples also show that IBIS issues are captured at different times in different local situations/contexts due to the ad hoc characteristic of the approach. Over time, the IBIS issues form a linear stream (see Figure 4.3) and explain the path to understand and solve the requirements problem.

![Figure 4.3: A linear stream of IBIS arguments is formed](image)

Although benefits of the IBIS record for software development projects have been reported in Conklin and Yakemovic (1991) and Yakemovic and Conklin (1990), the analysis was limited to the “the consistency in quality of note taking” when using the IBIS notation and the benefits become clear mostly when later reviewing the IBIS documents, specifically in detecting errors, tracking unsolved issues and understanding previous decisions when someone has left the project. This thesis has thoroughly studied the usefulness of the IBIS approach in describing the FOOM process. The research confirms, inter alia, that the ad hoc characteristics, the intuitiveness and the structure of the IBIS approach allow and support the description of the flow of ideas and changes to the requirements model. Although this conclusion has been achieved based on the use of IBIS in this project, it also strengthens and encourages research into the relevance and applicability of other ad hoc design explanation notations (see Chapter 2) in requirements engineering.
The description of the requirements engineering process, structured using the IBIS notation, also assisted the researcher to build a new understanding of the requirements engineering process. The IBIS record of the project describes decisions and changes to the model as time progresses. The critical decisions and significant changes to the model that were documented, show clearly that the process is not smooth but rather it involves occasional restructure and simplification of requirements models. A detailed analysis of the requirements process led to a new and exciting finding about the process. The new understanding and its implications for project managers monitoring the requirements engineering process will be discussed in detail in Chapter 6.

Communication between participants involves discussion about requirements in all three forms of FOOM specification. IBIS supports the preparation of the meeting’s structure, the connection of issues as discussed in different meetings, and the explanation of a specification. In this project, IBIS was used in all three of these ways. Often, unsolved issues about requirements questions (recorded in Issues and Sub-Issues without a selected Position) and proposed solutions (recorded in ?P) necessitate a meeting and form its structure. Issue, Positions and Arguments permit quick reference to decisions made or Positions left open in previous design meetings. IBIS also supports the explanation of previous decisions and the state of the requirements specification. This is very useful for the specifier and participants when discussing an Object-Z fragment. As a specifier I often started a modelling session with a list of open Issues. As we went along the session, we discussed ways to solve the problems and we recorded the deliberation.

There were many times when the participants structure IBIS arguments on whiteboard or paper, especially when dealing with complicated issues. The problem of involute link, described above, was discussed in several meetings. The difficulty of understanding and
representing this problem is due to the different abstraction levels and classification schemes involved:

- the Class and Instance concepts of object-oriented modelling: The class *Notation* represents an argumentation notation. The class *Argument* represents an actual argument which records rationale for a design decision. *Argument* is an aggregation of instances of the classes *Node* and *Link* which represent components of an argument. For example specific *Issue X* and *Issue Y* which are instances of the class *Node* may be components of an argument.

- the node and link types of argumentation notations: each (instance of) *Notation* defines a set of types for node components (*nodeTypes*), a set of types for link components (*linkTypes*), and a set of ‘legal’ connections between these *nodeTypes* and *linkTypes* for a particular notation. In IBIS, for example, *nodeTypes* is a set of *Issue*, *Position* and *Argument* components; *linkTypes* is a set of Objecting-to, Supporting, Generates, Responds-to, etc... components, where Objecting-to (or Supporting) connects a node of type *Argument* to a node of type *Position* and Generates (or Responds-to) connects a node of type *Issue* to another node of the same type *Issue*. Both the classes *Node* and *Link* have an attribute *type*. The attribute type of above *Issue X* and *Issue Y* is *Issue*. The *Notation* IBIS allows *Issue X* to be connected with *Issue B* through a link of type Generates or Responds-to. All the above definitions were recorded in different *Issues* when these classes were introduced and added to the model.

- the concepts of class and instance were understood differently by the various participants. This led to difficulties for the specifier in communication. Participant 2 often used the terms class and instance as they are used in object-oriented
modelling (the object-orientation classification scheme) while participant 1 often used type and instance to refer to his requirements, \textit{i.e.} argumentation components (the argumentation classification scheme). The two classification schemes may seem ‘parallel’. However, they are different, the object-orientation classification scheme categories components according to the representation role (\textit{i.e.} node and link) while the argumentation classification scheme categories these components according to the semantic role of an argumentation notation (\textit{i.e.} problem, solutions and rationale for solutions).

Therefore, the involute link could be understood at different levels. At the instance level of object-oriented modelling, the linkage of a specific \texttt{Issue X} to itself is not allowed in the IBIS notation. However, at the type level of argumentation notations, described above, Issues X and Issue Y can be linked: Issue Y may be a subissue of Issue X. These sophisticated concepts often led to confusion and misunderstanding between the specifier and two participants.

The problem was understood clearly through an example drawn from Entity Relationship Modelling (see supporting Arguments (AS) for Position 1 in Issue A). Therefore, this Position was selected when it was introduced initially. Considering different abstraction levels involved, participant 2 suggested a second Position for Issue A. This led to the rejection of Position 1 and a further exploration of the problem where it was finally solved (see Sub-Issue A1). Therefore, Issue A and the linked Sub-Issue A1 explain the Object-Z expression of the class Link.

Spelling out arguments and modelling ideas using the IBIS notation led to the explicit consideration of trade-offs involved in the choice between 'sticking' to the existing argumentation notations which do not allow involute links and having flexibility in
creating new argumentation notations with extendibility to use modelling notations. Abstracting from this discussion, the specifier and two participants generated a general issue related to the Argument for having flexibility. Later, this issue opens a new perspective of the requirements: a notation may be any notation, either argumentation notation or requirements modelling notation. Therefore, the resolution of the involute link problem and its explicit rationale were important for the project.

In summary, the IBIS notation is found to be very useful in reaching explicitly shared understanding. This finding confirms and also clearly explains what Conklin and Yakemovic (1991) and Yakemovic and Conklin (1990) refer to as “impact on Project Team Communication”. These authors claim that “the team meeting seemed to be more productive” (Yakemovic and Conklin, 1990, page 114) as a result of using IBIS.

The IBIS-structured documentation provides a useful and accessible information source for tracing and extracting concepts related to a requirements problem, especially when restructuring the requirements model. These concepts are expressed in Positions and Arguments. In this IBIS-structured documentation, there may often be a number of Issues related to a single requirements problem. For example, there are a number of Issues relating to definitions of and relationships between Decision-Model-Argument (see Figure 4.4(a)). These issues explain the changes made to the Object/Class model (from two binary to one ternary relationship) and the Object-Z specification of these classes. Issues B, C and D (above) are examples of such arguments. These Issues were retrieved and used in considering the two following modelling options:
The class Decision records a change from a Model (a version) to another Model (another version). The change is supported by a set of Arguments.

∀ arg : argument • arg.basedon = first(change) ∧ arg.concerns ⊆ first(change).structure ∪ second(change).structure

Figure 4.4: Two possible solutions were considered repeatedly
It follows that Issue B not only promotes better reasoning around different definitions of the class Decision, but also generates a Sub-Issue. Later the resolution of this Sub-Issue and other related arguments leads to a restructure of the requirements model (see Figure 4.5).

![Figure 4.5: Extract from specification version 6](image)

This demonstrates the usefulness of the IBIS notation in the provision of retrievable (re-traceable) working memory and results in the reduction of time and effort by the specifier. As demonstrated in the previous example, the IBIS base was used to trace back all arguments and issues that were related to a specific requirement. This is particularly important at crisis points of restructuring and simplifying the requirements model. This will be discussed in more detail in Chapter 6. It is also clear that the IBIS-structured documentation supports the audit trail of the evolution of a specific component of the requirements model e.g. an object or a class. In contrast to textual specifications, FOOM requirements can be split into discrete components. This characteristic is shared with other requirements engineering methods making use of “structured” models. Note that when clarifying a FOOM component, participants
often need to refresh their knowledge about previous decisions. The IBIS structure permits tracking down all decisions (selected Position: *P) and Arguments (AS and AO) that have been made among a number of related itIBIS arguments about the component. However, the searching for desirable information in the large IBIS base is time-consuming. Moreover, at the stage of problem understanding and structuring when the problem itself and the relevant concepts/ideas are not well-explored, the problem solver may not know exactly what to search for and whether or not it can be found in the base of previous decisions. A structure that conceptually summarises the IBIS base would be useful to the problem solver in such cases. This will be discussed, in some details, later.

Similar results have been reported in Conklin and Yakemovic (1991) and Ramesh and Dhar (1992). Conklin and Yakemovic (1991) focus on the role of IBIS in facilitating the detection of errors and the tracking of open issues when inspecting and updating the IBIS base following the conversion from itIBIS to gIBIS. Ramesh and Dhar (1992) also track the IBIS base through extending IBIS and using Telos language (see Chapter 2). This thesis focuses on the role of IBIS not only in tracking the evolution of requirements, but also in supporting and tracking the building and development of ideas and concepts while maintaining the simplicity and intuitiveness of the notation.

However, early in this requirements engineering project, some limitations of IBIS have also been found. First, the participants experienced what have been described by (Ramesh and Dhar, 1992) as the weakness of IBIS: its restriction of local context leading to a weak connection to the whole design. In our project, this weakness is observed and analysed at two different levels. Within the local scope of an Issue, the evaluation of different Positions is rather implicit. Although there may be a number of
Positions for a particular Issue, each Position is assessed in isolation by its own pro and con Arguments rather than by a common set of Arguments. For example, in Issue A, each Position (P1 and P2) is assessed by a number of its own supporting and opposing arguments (AS and AO). However, the AS and AO of Position P1 may be used to assess P2, in fact, they also partly support the specialisation of the class Link. At the global level, the IBIS Issues are structured around a local and partial problem as it occurs and with little consideration of the global systems requirements. For example, Issue B, C and C discuss different aspects of recording the requirements evolution and the rationale for the evolution.

Other authors overcome this weakness of IBIS both directly and indirectly. Some authors extend the notation with additional components which represent the client’s requirements and then link them to each specific Issue, for example see the adding of “Scenario” (Pries-Heje, 1993) and “Input for the task” in REMAP (Ramesh and Dhar, 1992). Other authors use a network of Issues with intention of representing procedural design knowledge though the serving relationship between Issues (McCall, 1986; McCall, 1991) (see PHI, Chapter 2) or of promoting project team communication (Conklin and Yakemovic, 1991). In the light of the above analysis of our experience, these previous approaches do not overcome the weakness of IBIS at the local level. They may enhance IBIS with the provision of the global perspective, however, most of them fail to provide a mechanism to structure and inspect the connection between specific Issues. This task may be daunting due to the quickly growing number of Issues in the design explanation base. The quasi-hierarchical tree of Issue in PHI may provide a mechanism for structuring related Issues, however, there are two questions: is the relationship ‘serve’ between Issues sufficient? and what should we do if the hierarchy
is not appropriate in structuring the relationship between issues? This analysis of the weakness of IBIS at two levels suggests that a mechanism to inspect the IBIS base and provide a global perspective of local issues is needed. Indeed, the need for reviewing the IBIS base raised at some specific times. Chapter 6 will further discuss this issue in detail.

Second, the locality of ad hoc arguments leads to difficulty in searching and extracting relevant and desired information from the large IBIS base. As illustrated in Figure 4.3, as time progresses, the IBIS arguments are accumulated and form a time ordered linear stream of issues. Although IBIS arguments are added to the base according to the time they are created, there is no better index than time to sort the issues. A desirable index would be problem-based, but the ‘problem’ is a tentative concept (the most appropriate classification of which is difficult to grasp) as problems might be named differently or overlapped. As the model grows, the IBIS base becomes larger and more disorganised. Searching for desirable information or concepts related to a specific requirements problem requires us to read in sequence the large number of all IBIS issues previously created and recorded in the IBIS base. Moreover, sometimes, an idea generated from an issue required revisiting previously issues, searched and discarded for this purpose. Therefore, although the IBIS base provides a useful and structured source of information, researching for information in this base is time consuming. This is depicted as a dotted line in Figure 4.1.

The difficulty in searching and retrieving desirable information from the IBIS base was also experienced by Conklin and Begeman (1988) and Conklin and Yakemovic (1991). Conklin and Begeman (1988) [page 306] developed the graphical hypertext software tool gIBIS for building and managing a large IBIS information base. Conklin and Yakemovic
used this tool in an attempt to overcome difficulty in managing the large IBIS base. We may, however, conceptualise this problem in a different and more useful way. Evidence shows that the linear stream of IBIS arguments tends to explain how things get there, however it is often not clear why in the end a particular specification has a certain form. I strongly believe that a global, holistic view is needed. Therefore, to overcome the limitations of the locality of IBIS and difficulty of searching in the IBIS base, I decided to construct a *post hoc* analysis around the *specification* rather then around a local problem. This analysis would be constructed at a separate time, at intervals between phases of the active modelling process, rather then during the modelling process. The subsequent research cycle is conducted with a view to supplementing *ad hoc* with *post hoc* design explanation within requirements engineering, and is reported in Chapter 5.

Lastly, the participants also experienced problems due to a lack of explicit context in the IBIS notation. IBIS argumentation does not document the context in which an *Issue* is generated. The knowledge of the context surrounding the original debate about an *Issue* is desirable when reappraising the *Issue* (these events may be separated by a considerable time) and is important when examining the *Issue* in relation to other *Issues/Sub-Issues*. To better describe the *Issue*, I have added a description of the “*Context*”. Therefore, each IBIS fragment describes a detailed situation (*Context*) and why (*Argument*) a decision (*Position*) is taken to address a problem (*Specific Issue*). See above *Issues A, B, C and D* for examples. The use of the additional element will also be further examined in Chapter 5.
4.3 Using IBIS more effectively

4.3.1 A summary of findings

In using the IBIS notation to structure the requirements evolution and to record the rationale for the evolution, the findings

- highlight two types of arguments,
- suggest attaching each argument to particular FOOM components,
- suggest an investigation into the use of a post hoc approach of design explanation in a complementary manner to IBIS,
- confirm the advantages of having a specifier (not a scribe) to use IBIS.

4.3.2 An analysis of findings

During the project, intermediate versions of the requirements model and changes to the versions were also documented (see Figure 4.6. The IBIS documents were attached to the intermediate versions and associated changes. We found that there are two types of arguments: confirmation of a version and confirmation of a transition (which may be a rejection of some parts of the current version). Issue C and D (see the previous section) are two examples of these two types.

Issue C confirms the current solution at the time when the Issue was generated (see Figure 4.4(a)). However, when returning to this Issue in another Issue (Issue D), in order to record static state, changes and dynamic transitions of Model, the participants decided to have a ternary relationship. This suggested changing our requirements model.
These findings suggest a need to specialise the class Decision. Two types of design decision are Documentation, containing arguments about the appropriateness of a Model, and Transition, holding arguments for a transition from one version to another. Potts and Takahashi (1993) specialised the effect of an argument on a requirements element into five categories: refinement, clarification, merge, split and retraction. However, our simpler classification has been found appropriate and is adequate to structure the evolution graph of requirements specification versions (see Figure 4.7).

We attach our arguments to particular components involved in a specific change rather than to the whole of a FOOM model as originally proposed. Indeed, Issues A, B, C
and D are related to a part of the specification (see Figure 4.4 and associated Object-Z fragments) rather than to the whole specification. There are also other issues related to the maintenance of semantic mappings in FOOM, e.g. one real-world object appears in different models within a FOOM specification (again this situation is common to most model-based requirements methods). The integration of the IBIS arguments into specific requirements (design objects) also enhances the traceability of FOOM components.

The use of IBIS in relation to version control must be examined carefully. For example, the specifier traces back all issues related to a part of the model involved in the transition between two versions. These issues can be divided into different groups representing different areas of concern, e.g. how to represent the relationship of \textit{Decision}, \textit{Model}, and \textit{Argument}, how to specify methods of \textit{Project} and \textit{Argument}, and how to specify the constraint in the Object-Z specification of the class \textit{Decision}... Every single change does not require a transition to a new version. Two important questions concerning version control arise:

- when is it sensible to release a new version? At the beginning of the research every decision generated the release of a version. However, the number of versions of Object-Z models grew very rapidly compared to the number of versions of structure diagrams. We decided that the concepts of specification, model and version need to be further clarified.

- do we need and how can we structure a representation of the current logical space to assist us to evaluate two versions? There is a common dilemma: whether to keep a current version or to release a new version. As mentioned previously, the restricted locality of IBIS \textit{Argument} leads to difficulty in comparing FOOM
specification versions. So, how do we evaluate collections of arguments related to different specification versions?

Again, these questions also indicate that the sole use of IBIS may not be adequate for an effective use of explanation information within the requirements engineering process. A post hoc logical analysis might be used to complement IBIS. This strengthens the suggestion for the second research cycle.

Having a specifier to use IBIS assists the specification team in:

- achieving a close connection between the specification artefact and the explanation information.
- avoiding possible misinterpretation, which I believe would occur to a much higher degree if an outsider-scribe was used (i.e. we are being more confident with data collected). Indeed, the interpretation of facts, especially the description of the context via the element “Context”, reflects a rather subjective view on the part of the specifier.

### 4.4 Conclusion

In conclusion, this research cycle suggests that design explanation is useful within the requirements engineering process. It has also revealed ways in which IBIS might be used more effectively. The FOOM requirements evolution and the rationale for the evolution are documented using the IBIS notation. Specifically, discussions and decisions are recorded as they occur and explanation documents written in the extended IBIS notation (itIBIS with “Context”) and are attached to the requirements. The
participants’ experience in using IBIS indicates that this approach would improve the
decision making activity and the understandability of requirements specifications and
support communication between different participants. In consequence, this would
increase confidence in the conformance of the specification to the user’s requirements,
and in its consistency and appropriateness.

Nevertheless, there are also a number of limitations of IBIS. Specifically, they include
the lack of a holistic view in the IBIS argument and the difficulty in searching for
a specific problem from a large number of local IBIS arguments. To overcome these
limitations, further research will be conducted with a view to investigate the benefits
of supplementing the ad hoc IBIS arguments with additional rationale structured using
a post hoc design explanation notation.
Chapter 5

Supplementing ad hoc IBIS with post hoc QOC

In the previous chapter, I described the first research cycle in which the ad hoc design explanation notation IBIS was used to record discussions and decisions about FOOM requirements as they occurred. Having reflected upon this application, I concluded that IBIS has both benefits and limitations, and suggested an exploration into supplementing IBIS using a post hoc design explanation notation. These conclusions form the basis for the second research cycle.

In this chapter, I describe the second hermeneutic cycle of the research programme. A brief introduction to the research cycle has been provided in Chapter 3, Section 3.3.3. The components for this research cycle are:

**Intellectual framework** (F), derived from the discussion in Chapter 2, has been strengthened and further refined with the finding from the first research cycle (see Chapter 4). Specifically, it is that the systematic documentation of the
requirements evolution process and the underlying rationale may be used both to describe and explain the requirements engineering process and also to improve the understandability and traceability of the requirements specifications. Further understanding of the documentation process is needed.

**Method (M)** As suggested by the reflection upon the use of IBIS during the previous research cycle, the method being explored during the second research cycle was to use QOC to supplement IBIS within FOOM.

**Real world application (A)** The specification of NGAW, written during the first research cycle, supported the switch of argumentation notations and the attachment of explanation information to a specific design object. Furthermore, the usefulness of structuring the evolution graph of specification versions and attaching the IBIS arguments to the graph led to the extension of NGAW. The project became the specification of requirements for a FOOM super-structure CASE tool to manage the evolution of the FOOM requirements. This will be described in more detail in Section 5.1.2.

The research cycle is described in this chapter in the same manner as in Chapter 4. The research process is described and the qualitative findings are analysed. The analysis will focus on the following issues: whether both QOC and IBIS are useful and how the two notations can be used together effectively. Particularly, I focus on whether and, if so how, QOC overcomes the weaknesses of IBIS found in the first research cycle.
5.1 Research process: observation and data collection

This section explains why QOC has been chosen to overcome the limitations of IBIS, it outlines the extended requirements engineering project as a real world application for the study, and it describes the process of using both IBIS and QOC within the requirements engineering process.

5.1.1 Choosing a process-oriented design explanation notation

In choosing an additional design explanation notation to overcome the limitations of IBIS, I have considered the following issues:

**Relevance** IBIS arguments are often focused at a local, detailed level, each argument being independent of the others and not formally related to form a holistic picture. The limitations of the focus/locality of IBIS and the difficulty of searching in the IBIS base suggested the researcher supplement the process-oriented IBIS base with additional *post hoc* rationale constructed around the *product* rather than around a specific problem encountered during the process. Therefore, the notation to be chosen must necessarily support a *post hoc* approach to design explanation.

**Simplicity** Because two design rationale approaches were desirable and two notations would be used in tandem within FOOM, a *post hoc* notation which is easy to learn and use and yet sufficiently expressive was required. This is especially important because the additional construction of *post hoc* rationale is, by its nature, somewhat intrusiveness.
Among a number of the design explanation notations discussed in Chapter 2, QOC (MacLean et al., 1991b) and DRL (Lee and Lai, 1991) are two notations that can be considered as supporting a post hoc approach to design explanation. Indeed, having been designed to be a structure-oriented notation, QOC supports the post hoc reconstruction and review of the design space around the design product. In addition, with its emphasis on the richness of a fully developed representation language, DRL also emphasises the logical (rationalised) design explanation while other notations tend to focus on only the chronological design explanation (see Chapter 2 for details).

However, these notations indicate different emphases. Shum (1991a, page 263) judges that QOC, created either during or after actual design meetings, aims at providing an abstracted rationale, which outweighs the loss of narrative characteristic while DRL, though often created after actual design meetings, tends only to provide the “narrative rationale about local issues” (Figure 5.1).

With regard to the simplicity issue, DRL, with the richness of its vocabulary and the soundness of its semantic reasoning, is however considered as rather complicated. Therefore, when being used complementarily with IBIS, this notation would require a lot of additional effort on the part of the requirements engineer.

On the other hand, QOC is both simple to learn and to use. As the discussion in Chapter 2 demonstrates clearly, QOC is also an adequately expressive post hoc notation. It is also clear that is popular and a major focus of research into design explanation. There has been a growing number of intensive research projects into developing and enhancing QOC mainly at Rank Xerox Cambridge EuroPARC. There have also been numerous studies evaluating QOC as well as attempting to integrate QOC with other design methods. QOC was found to be useful in constructing design space analysis
and solving critical problems, but should not be considered to form a full description of design space.

There was another important issue in choosing additional design explanation for the second research cycle. The focus of this study is the exploration of the possibility of supplementing the *ad hoc* with the *post hoc* approach to design explanation within requirements engineering. Since the study emphasises the research issue at the conceptual level, an existing representative *post hoc* notation was chosen rather than developing a new *post hoc* notation (or using IBIS in a *post hoc* manner).

In summary, QOC was chosen in our study due to its relevance, simplicity, expressiveness, popularity and usefulness.
5.1.2 The requirements project—an extended application

Having taken into account the learning from the first research cycle, the focus of analytic effort, Notation Generic Argumentation Workbench was revised and extended to be a FOOM process management CASE tool. Requirements of this CASE tool were far more complex and larger than expected and much more complex than the NGAW of the first study cycle. This would be a tool to manage the evolution of a FOOM specification and to record the design explanation relating to this evolution. Therefore, learning from the action research study also directly contributes to the understanding and improvement of the requirements. The extended project took over 12 months with the same staff and conditions similar to these described in Chapter 4. A description of the project is provided below.

The FOOM requirements evolution process is viewed as a network of specification versions. As shown in Figure 5.2 (derived from the previous research cycle), each version is an intermediate state of a FOOM specification at a specific moment (static state). These versions are linked together by transitions (dynamic evolution) according to the development process. One of the requirements is to keep track of these intermediate states of a FOOM specification and the transition from one to another state using an approach similar to Resource Control Systems (RCS) or Source Code Control System (SCCS). Another requirement is to attach decisions and arguments into this model of evolution to explain why each version has a certain form and why a transition occurs. The CASE tool has to support monitoring of the FOOM process, provide explanation and therefore improve understandability and traceability of the specification to both the analysis team and, later, to the design and implementation teams.
5.1.3 Using QOC and IBIS within the FOOM process

During the second research cycle, all requirements arguments were recorded and coded using either IBIS or QOC. They were attached to a particular FOOM model or a transition under discussion. The process of using the two notations was observed. Figure 5.3 illustrates what happened.

As described in Chapter 4, the requirement engineering process is cyclic with each cycle consisting of eliciting, modelling and evaluating requirements. The creation of the IBIS arguments remains the same as in the previous research cycle: in the modelling activity, arguments about requirements were recorded and structured in IBIS as they were made. These IBIS arguments were a good source for structuring logical analyses in QOC. In the evaluating activity, QOC analyses were used in isolation for the evaluation of different candidate solutions represented in FOOM models (Object-Oriented diagrams and Object-Z). Unsolved or incomplete IBIS arguments and QOC analyses lead to further requirements elicitation. IBIS and QOC documents were also used in communicating with other participants in two ways: for assessment of requirements
criteria for validation of solutions, and for further elicitation of old or new requirements.

Therefore, as the time progressed, requirements were explored, discussed and structured, and our discussions were captured and documented using the IBIS notation. They were attached to the associated versions of the model as described in Chapter 4 Figure 4.1. In this research cycle, IBIS issues were converted into a QOC analysis (Figure 5.3) for the evaluation of modelling options from time to time. Both the IBIS and QOC documents were attached to the evolution network of the requirements model. Figure 5.4 illustrates the process of documenting the evolution of both FOOM models and design explanation documents.

There are two important characteristics of using design explanation during this research cycle:

- The specifier was also the creator of the IBIS arguments and QOC analyses. This approach is supported by benefits of the ‘direct use’ approach to creating and using design explanation by the specifier, found from the previous research cycle (see Chapter 4).
The QOC post hoc analyses were structured and represented using graphical diagrams as described in MacLean et al. (1991b). This decision was supported by findings from Shum’s (1993) study that graphical diagrams are preferable among various QOC representation forms. The IBIS ad hoc arguments, created during actual requirements modelling process, were continually recorded using the textual itIBIS form in order to support non-intrusiveness and to minimise any interference in the process.

Apart from what is described in Chapter 4, additional activities of the researcher during this research cycle included the conversion of IBIS arguments into QOC analyses through the post hoc reconstruction of the design space using QOC where the IBIS base serves as a source of information. QOC analyses were constructed when the following situations occurred:

- The IBIS base grew inconveniently large. Access and retrieval to the IBIS issues became difficult. Clearly, there was the need to reorganise the IBIS base.
• A critically difficult problem was encountered, for example different issues related to the problem or viewing the problem from different perspectives prompt different or contradictory positions.

• The specifier felt the need to create QOC analyses intuitively.

QOC analyses represented in graphical diagrams also served as research data in addition to research data described in Chapter 4. Overall, research data included:

• FOOM requirements discussions and modelling decisions structured in the IBIS and QOC notations. More than 430 issues and 20 QOC documents were recorded in the research diary.

• Questions and issues concerning the use of both IBIS and QOC within the requirements engineering process documented in the form of notes or issues without positions and arguments. These also included post hoc semi-structured interviews of two participants at different stages. In addition, the researcher/specifier kept taking notes on her thought processes.

• Intermediate states of the FOOM models (object-oriented diagrams, Event Chain diagrams and Object-Z diagrams).

The data were qualitative and provided a rich and meaningful picture of the participants’ reasoning and their actions. Although the supplementing of IBIS with QOC was the primary focus for the analysis of data collected, findings from the previous study were also re-examined and refined. The next section will discuss whether both QOC and IBIS are useful, and how the two notations can be used together effectively. Particularly, it analyses if and how QOC overcomes the weaknesses of IBIS found in earlier
research. Issues raised during the research cycle, together with issues highlighted in Chapter 4, form a basis for the discussion in Chapter 6.

5.2 The usefulness of IBIS and QOC

5.2.1 A summary of findings

Overall, evidence from this study confirms the findings of our previous research about the usefulness of IBIS. Furthermore, the two notations provide a well-structured and traceable description of the FOOM process, explain FOOM specifications and assist the communication of ideas between participants. Particularly, analysis of observation data shows that design explanation information:

- provides the FOOM specifier with both a chronological record of requirements arguments, structured in IBIS, and a logical restructure and evaluation of requirements models, structured in QOC,
- improves understandability of the FOOM models represented in various forms of semi-formal diagrams and formal documents in Object-Z,
- reduces searching time in the large IBIS base,
5.2.2 An analysis of findings

**Providing both chronological record of the requirements engineering process and a logical evaluation of requirements models**

Throughout the process, IBIS arguments were used to record discussions about a specific requirements problem. As time progressed, the IBIS arguments formed a linear flow of issues which was useful for the description of the FOOM process. However, there are limitations of the IBIS arguments as identified from the previous cycle. First, the IBIS base grew rapidly, thus the accessing of relevant information in the base was difficult. Second, each IBIS argument records only a specific, local requirements problem, thus, it does not provide a holistic picture of the problem situation and the solution state. This subsection will demonstrate how a QOC analysis was solved these limitations of IBIS arguments.

IBIS was confirmed to be very suited for recording the ongoing process of either confirmation of a selected model or making changes to that model. Indeed, IBIS arguments show that the modelling process tends to be based on a selected specification version and involves discussions about a part of the version until a relatively stable and satisfactory specification is achieved. The following is an example. As the workbench was extended to support a specific requirements engineering method—FOOM, the definition of the classes *Project* and *Model* had to be revised. Figure 5.5 is a part of version MAW7.2 which was selected for a debate about a definition of the FOOM specification version and its relationships to *Model*, *Documentation* and *Transition*. At that time, each *Documentation* was attached to a *Model* and each *Transition* was attached to two *Models*—the from and to sides of the transition. Each *Project* was a graph of
Models. Each Model represented a snapshot of the FOOM specification at a specific time. There were a number of issues related to this debate. Among them, Issues E, F and G record important discussions between the FOOM specifier and her participants about this matter.

![Figure 5.5: A part of version MAW7.2](image)

Issue E developed a definition for the FOOM specification which was refined in Sub-Issue E1. It is evident that this IBIS issue is useful not only in recording the discussion but also in clarifying our understanding:

**Specific Issue E** Specifier: What is the definition of a FOOM specification version?

**Context** The Specifier and Participant 2 discussed about version control for different forms of a FOOM specification, text, structure, communication and Event Chains diagrams and Object-Z specification. Previously, in the specification for NGA, each version is modelled
as a *Model*. Each *Model* is a node-link diagram. However, as the application becomes a FOOM CASE tool, this definition of a FOOM model as a version is questioned.

*P 1* There is the need for a class to co-ordinate all kinds of different diagrams for a FOOM specification version—*Specification*

**AS** Different diagrams evolve differently. Structure diagrams are more stable than others. Object-Z documents evolve most, probably because there are operations involved in the specification of a class. Event Chain diagrams evolve more than structure diagrams because there are many scenarios possible for a structure diagram.

**Sub-Issue E1** In this case, do we want to record the evolution and the rationale (for the evolution) of both a complete specification and its component models?

*P* Yes, we want to document the evolution and the rationale of changes to requirements at both levels of a complete specification and component models.

Issue E led to the creation of the class *Specification* and pointed out a need for documenting the evolution of requirements at two levels of the complete FOOM specification and component model. The creation of *Specification* led to an important change to the selected structure diagram. New problems included:

- how to specify relationships between the new class and other existing classes in the current diagram which were taken up in Issue A
- how to equate the previous definition of *Model* as a version of the new class *Specification* taken up in Issue A, and as a node-link diagram in its own right

Being constructed from the second perspective, **Issue F (below)** generated three Positions for consideration, see Figures 5.6(a), 5.6(b) and 5.6(c) of which the third was rejected
and the others were left undecided.

**Specific Issue F** Do we need a relationship between *Specification* and *Decision*?

**Context** According to the outcome of Issues A2-01-201196 (Issue A) and A2-I2-071296 about the FOOM specification version, each FOOM project is seen as an acyclic graph of *Specifications*. The class *Specification* represents the FOOM specification version; each instance of the class *Specification* contains a set of different FOOM documents—*Model*. Since each design decision consists of a number of arguments about a *Model* or a transition between two *Models*, the relationship between *Argument* and *Specification* has become difficult to understand.

?P 1 There must be an additional explicit relationship between subclasses of *Decision* and *Specification* (Figure 5.6(a)).

```
Documentation
<table>
<thead>
<tr>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>model : Model</td>
</tr>
<tr>
<td>forSpec : Specification</td>
</tr>
</tbody>
</table>

∀ arg : args •
  arg.concerns ⊆ model.content

Components recorded in arguments confirming a Model must be components of the Model. *m.content* is a set of all Nodes and Links of the model *m*.

model ∈ forSpec.models

The model must be one model of the specification.
```
**Transition**

**Decision**

\[ \text{fromTo} : \text{Model} \times \text{Model} \]

\[ \text{forSpecs} : \text{Specification} \times \text{Specification} \]

\[ \text{first}(\text{fromTo}) \neq \text{second}(\text{fromTo}) \]

It is necessary to keep the evolution graph acyclic.

\[ \forall \ arg : \text{args} \bullet arg.\text{concerns} \subseteq \text{first}(\text{fromTo}).\text{content} \cup \text{second}(\text{fromTo}).\text{content} \]

Components recorded in arguments for a transition must be components of only the pre-existing and the newly created Models.

\[ \text{first}(\text{fromTo}) \in \text{first}(\text{forSpecs}).\text{models} \land \]

\[ \text{second}(\text{fromTo}) \in \text{second}(\text{forSpecs}).\text{models} \]

The pre-existing model and the newly created one must be models from the pre-existing and newly created versions, respectively.

\[ \ldots \]

**AS** To express explicitly, for example, how arguments support each Specification and the transition from one Specification to another explicitly.

**AO** Information about design decisions and Specifications could be derived from the relationships Documentation − Model, Transition − Model and Specification − Model.

**AS** To attach design decisions to a Specification and transition between them directly.

**AS** To record evolution of the FOOM specification of a whole (in Specification).

**AO** It is derivable information, could be derived by an operation.

**AO** Faster than if derived by operation during the run-time.

**P 2** Redefine the class Documentation, Transition and Arguments (Figure 5.6(b)).
Decision

\[ \text{args} : \mathbb{P} \text{Argument} \]

...

Argument

Model

\[ \text{concerns} : \mathbb{P} \downarrow \text{Node} \]
\[ \text{models} : \mathbb{P} \text{Model} \]

\[ \text{concerns} \subseteq \bigcup \{ m : \text{models} \bullet m.\text{content} \} \]

\[ m.\text{content} \text{ is a set of all Nodes and Links of the model } m \]

....

Documentation

Decision

\[ \text{forSpec} : \text{Specification} \]
\[ \forall \text{ arg} : \text{args} \bullet \text{arg.models} \subseteq \text{forSpec.models} \]

\[ \text{specification.models} \text{ is a set of all FOOM models of the specification } \text{forSpec}. \text{ All arguments in a Documentation must be about models of the same version.} \]

....
<table>
<thead>
<tr>
<th>Transition</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fromTo : Specification × Specification</code></td>
<td></td>
</tr>
<tr>
<td>( first(fromTo) \neq second(fromTo) )</td>
<td></td>
</tr>
</tbody>
</table>

necessary to keep the evolution graph acyclic.
\[
\forall arg : args \bullet arg.models \subseteq first(fromTo).models \cup second(fromTo).models
\]

All arguments in a Transition must be about transition of models involved in the same transition between its two specifications.

...  

**AS** Design decisions are sets of arguments and are attached to different specification versions and transitions between them. Each argument in a design decision, *Documentation* or *Transition*, must argue about models of a specification or transition between them respectively.

**AO** *Transitions* between *Models* are implicit.

**-P 3** Keep *Documentation – Model* and *Transition – Model* and add Object-Z operations to retrieve information of how subclasses of *Decision* support specifications and their transitions (Figure 5.6(c)).
Project

AcyclicProperGraph

\[
\begin{align*}
decisions : & \mathbb{P} \text{Decision} \downarrow \\
\text{network} : & \text{Specification} \leftrightarrow \text{Specification} \\
\forall d : & \text{Documentation} \bullet d \in \text{decisions} \Rightarrow \\
& \exists \text{spec} : \text{Specification} \bullet \text{spec} \in \text{dom network} \cup \text{ran network} \land \text{d.model} \in \text{spec.models} \\
\forall t : & \text{Transition} \bullet t \in \text{decisions} \Rightarrow \\
& \exists s_1, s_2 : \text{Specification} \bullet (s_1, s_2) \in \text{network} \land \\
& \text{first}(t) \in s_1.models \land \text{second}(1) \in s_2.models \\
\end{align*}
\]

This is to ensure that all decisions of a project must support only

specifications and transitions of the project

---

_retrieveDocumentations

\[
\begin{align*}
aSpec? : & \text{Specification} \\
docs!: & \mathbb{P} \text{Documentation} \\
aSpec? \in & \text{dom network} \cup \text{ran network} \\
\end{align*}
\]

We can select only an existing Specification

\[
\forall \text{doc} : \text{docs!} \bullet \text{doc} \in \text{decisions} \land \\
\text{doc.model} \in \text{aSpec?.models}
\]

Any Documentation retrieved from the set of decisions must

confirm a model of the selected Specification

---

_retrieveTransitions

\[
\begin{align*}
aSpecTran? : & \text{Specification} \times \text{Specification} \\
trans!: & \mathbb{P} \text{Transition} \\
aSpecTran? \in & \text{network} \\
\end{align*}
\]

We can select only an existing transition between one to another

Specification

\[
\forall \text{tran} : \text{trans!} \bullet \text{tran} \in \text{decisions} \land \\
\text{first(\text{tran}.fromTo)} \in \text{first(aSpecTran?).models} \land \\
\text{second(\text{tran}.fromTo)} \in \text{second(aSpecTran?).models}
\]

Any Transition retrieved from the set of decisions must argue

for the selected transition between one and another Specification

---
AO Slow run-time

AO The relationship between design decisions and different versions is implicit in the structure diagrams.

![Diagram](image.png)

(a) Position 1  
(b) Position 2  
(c) Position 3

Figure 5.6: Three possible positions for Issue B

Being constructed from the second perspective (of equation of Model to Specification), Issue G defended the Position 5.6(c) which was rejected in Issue B. Issue G, however, rejected the Position 5.6(b) which was left undecided by Issue B. It is obvious that Issue G argued in favour of not changing the relationships.
Specific Issue G  How to equate the previous definition of *Model* as a version to the newly defined more or less complete FOOM *Specification*?

**Context**  As mentioned in the context of this issue, each version was previously a requirements model. After the argument on versioning FOOM specifications, each FOOM version—*Specification* becomes a set of FOOM documents/models, either in the text form, diagrams or in Object-Z—*Models*. This discussion is about the position of *Specification* and *Model* with other classes in the current specification.

*P 1*  All relationships remain the same. Only *Specification* is inserted between *Project* and *Model*.

**AS**  *Decisions* and *Arguments* explain changes to each specific *Model* directly rather than a set of *Models* as a consequence. Different forms of FOOM models are semantically linked and can be checked mechanically.

**AS**  There is a need to keep track of transition between each form of the FOOM specification.

-P 2*  All previous relationships between *Model* and other classes concerning the definition of *Model* (as a graph of node and links and as the superclass of *Argument*) remain unchanged. However, the relationship between *Documentation/Transition* — *Model* is replaced by the relationships *Documentation/Transition* — *Specification*, respectively.

**AS**  There is a need to keep track of transitions between specifications as a whole.

**AO**  Then how to trace back to which FOOM component/model involved in a particular change? This information is lost because there is nowhere to record the model(s) under discussion.

There were also a number of other related *issues* about definitions of *Specification*, the relationship between *Specification* — *Model*, the description of the FOOM CASE tool browser and how to document changes to versions, models and components. Indeed,
Issue E was reconsidered and lead to another Issue which clarifies the relationship between Specification and Model: aggregation vs. association. This issue was not solved at first, it came back several times during the modelling activity. The participants’ attention was switched between this Issue and attempts to specify Event Chains diagrams for a FOOM browser. A number of scenarios were constructed for understanding the FOOM browser in relation to recording discussions and changes to a FOOM requirements specification. It seemed that the representation of the relationship between Specification, Model, Documentation and Transition would be critical in order to progress further. Attention was switched back to this Issue. Before moving on from here, let me reflect upon the use of IBIS so far.

The IBIS Issues formed a linear stream of issues considered during the FOOM process in which an issue often leads to another. In fact, they represented a flow of FOOM modelling activities (see also Figure 5.3) which ranged from the creation of the class Specification to understanding different possible specifications of the relationship between this class and Model, Documentation and Transition. Each Issue addressed one aspect of the change to the definition of the FOOM version. During the modelling, Positions were discovered and elaborated as new IBIS arguments were recorded. Therefore, a group of some IBIS arguments represented a chronological flow of requirements discussions on some related topic. Each argument either confirmed a selected model or recommended changes to it. While the Issues recorded form a linear stream in respect to time, the logical path of analysis is rather chaotic. The focus of the cognitive design process alternates:

- within different problem areas, for example the different aspects of the problem (above), and at the same time
• within different levels of abstraction including, for example, the structural dia-
gram, the detailed specification of classes/objects, and occasionally implementa-
tion issues.

This observation is consistent with Carroll and Swatman’s (1998) description of the
requirements engineer’s traversal of the problem space. The observation will be further
analysed in Chapter 6.

The IBIS information was very desirable and valuable to the FOOM participants, es-
pecially for the specifier, in exploring and examining different positions carefully, in
preventing losing ideas, in spelling out arguments and in recording alternative changes.
The above issues serve as clear examples. In addition, the IBIS base is also useful
in refreshing previous discussions between participants. Indeed, during the modelling
process, participants often asked the specifier why a previous, certain modelling choice
was made. For example, “Why do we represent an Object-Z specification as an instance
of the class Model, i.e. a node-link diagram?” and “What does the class Specification
(Version) represent?”. There were also questions related to issues from the previous
research cycle, for example, “Why did we select an acyclic structure for the evolution
graph?” or “Why is an involuted link from a particular argument’s element to itself not
allowed?”. The usefulness of IBIS for communication of ideas is also confirmed: it was
often used to sum up a discussion to avoid misunderstanding between participants.

On one hand, the usefulness for the description of the FOOM modelling process of IBIS
is confirmed. On the other hand, IBIS also showed its limitations identified in earlier
research. They were carefully examined as follows.

The limitations lie in the implicit evaluation of positions and the locality of issues. Due
to these limitations, the assessment of different Positions within and across Issues is very hard. Firstly, the fact that each Position is assessed by its own set of pros and cons leads to the rather implicit evaluation of different Positions. Issues F and G show clearly that the resolution of an Issue cannot be solved merely because of numbers of pro and con Arguments for each Position. Secondly, Positions may be expressed and/or solved differently in different Issues. However, they may represent the same modelling option. The third Position of the Issue B and the first Position of the issue C illustrate such cases. They were assessed differently while representing the same option of Figure 5.6(c). Moreover, an Argument may address a Position differently in different Issues and contexts (versions). For example, the Argument which supports the definition of Model as a subclass of Node is no longer valid after the creation of the class Specification. This confirms the importance of the newly added element Context.

As argued in Chapter 4, in order to overcome this limitation it was desirable to restructure logically and evaluate accurately different alternative requirements models against a common set of criteria in a higher degree of abstraction and from a point of view wider than IBIS arguments each has. We decided that this is when a post hoc analysis is needed.

Figure 5.7 represents a QOC argument constructed for the review of the above group of Issues. The question in this QOC argument represented a requirements problem of the group of IBIS arguments. Analysis of these IBIS arguments led to three possible options to the question of how to document the evolution of a FOOM specification/model. With QOC, all three options were evaluated against a common set of criteria. Therefore, all these options were evaluated explicitly and accurately. These criteria were derived from all Arguments supporting and objecting to Positions of IBIS Issues. However, rather
than representing a concrete argument, each Criterion represented a requirement at a higher level of abstraction. From our experience, the evaluation of an Option against a Criterion can be explained by an Assessment Argument (MacLean et al., 1991b) which, in the project, was often an Argument in an IBIS issue. Putting it in other words, Criteria represent clients’ requirements while Assessment Arguments explain why a certain Option supports a Criterion. The assessment Criteria may include both specific (for example C1, C2, C3) and general criteria (for example C4 and C5), thus is useful in evaluating the specification at different stages of development. We find this organisation of Argument and Criteria manifests a better link between clients’ requirements and specifications.

Therefore, QOC arguments provide logical analyses around a specification while IBIS arguments provide a historical record of the specification process. However, there are three questions: does QOC come at a cost of the creation of QOC? are there overlaps
between the IBIS and QOC information? and when best to stop recording IBIS and do QOC? In the above example, the specifier and her participants did QOC when they noticed that it was hard to sort things out from a number of IBIS issues containing different and contradictory arguments. These questions will also be discussed later (in Section 5.3). To answer these questions, we first need to examine and understand the usefulness of QOC more carefully. Indeed, the usefulness of QOC will be further illustrated and analysed in another example in Chapter 6, which inter alia discusses the complementary use of IBIS and QOC in the light of the new understanding of the requirements engineering process.

Providing a unified view across different views on requirements

The attachment of some Object-Z fragments and/or object-oriented diagrams to IBIS Positions was confirmed. This attachment was useful in understanding different requirements models. These models reflect different views (perspectives) but they all concern the same underlying system. Consequently, these requirements models may evolve in parallel while still maintaining some semantic correspondence with each other (see Figure 2.15 in Chapter 2). This is a rather complex process and the understandability of a requirements specification tends to be problematic. As discussed in Chapter 2, this is a generally accepted difficulty in current requirements engineering practice. In this project, design explanation and their associated diagrams and Object-Z fragments assisted us in providing the participants with a unified view across different representation forms.

The attachment of design explanation to different requirements models was useful in two ways. First, relevant fragments of the specification to a Position were available when
needed (see for example, Figures 5.5 and 5.6 and the Object-Z fragments in Issues F and G). Second, supporting and objecting Arguments for this Position assisted us in understanding and consolidating different views on a particular requirement. For example, various Arguments of Issues F and G improved understandability of modelling options viewed from both the structural (see Figures 5.6(a), 5.6(b) and 5.6(c)) and formal perspectives (see the associated Object-Z fragments in Issues F and G). The structural diagrams are often preferred for their intuitiveness, thus offer users a quick grasp of an issue. However, the diagrams are still merely semi-formal, thus lack precision and may accommodate different interpretations. The associated Object-Z fragments provided additional information which clarified our understanding of the diagrams. However, it may not be understood quickly and easily by non-developer participants. Let us examine the following example.

Diagram 5.6(a) shows two ‘similar’ sets of relationships between two subclasses of Decision and Model and Specification. After an Object-Z specification was provided, did it became clear that these relationships work only at the Model level. According to the Object-Z fragment in Position 2, each instance of Documentation keeps Issues (Argument) about one Model of a Specification; and each instance of Transition keeps Issues about one transition from one to another Model, consequently from one to another Specification. Therefore, there may be a number of instances of Documentation for a Specification and similarly, a number of instances of Transition for a transition at the Specification level.

After this became clear, one of the participants added the need for having just one Documentation for a Specification as a whole and one Transition for a Specification-Specification transition. Therefore, Position 2 was proposed: each instance of the class
Decision keeps all the arguments about a Specification as a whole or a transition at the Specification level (see Figure 5.6(b)). Each Argument would store additional information about what Model it supports. This Position is assessed with Arguments for and against it (see Position 2 and the associated Object-Z fragment in Issue F). The Position was supported thought it wasn’t decided what Position would be the ‘best’ solution.

Reviewing this problem from the implementation perspective, another participant suggested keeping Decision at the Model level because the ‘primary’ modelling activities often involve the structuring and modifying of a specific model rather then a specification as a set of models. Design explanation documents at the Specification level was suggested to be retrieved through Object-Z operations. In summary, each Position and its associated Arguments consolidate both structural diagrams and their associated Object-Z specifications and highlight the reasoning behind the modelling option.

The usefulness of design rationale in providing a unified view across different views on requirements is important because the literature often reports difficulties in communication between stakeholders and in understanding Object-Oriented diagrams and formal specification experienced by the users-participants (see Chapter 2). A number of authors report the usefulness of design rationale in improving the understandability of formal specifications and in summarising and generalising specification issues (see Chapter 2). However, they tend to see the specification as a single requirements model. In fact, most requirements engineering methods involve different views of requirements, expressed in different representation forms and evolving in parallel over a long period of time. For example, in Booch’s Object-Oriented approach (Booch, 1994), requirements can be expressed as static structure models and/or interaction models. The correspondence between different representation forms of requirements in their parallel evolutions
can be related to the concept of vertical traceability introduced by Ramesh and Edwards (1993) (see also Chapter 2). The finding, therefore, strengthens our confidence in the use of design explanation in supporting this complex process of requirements evolution.

In addition, design explanation was useful not only for the specifier in communicating the modelling options to other participants, but also for the participants in validating the options, who made contributions by adding and editing Arguments (and Criteria in QOC) and negotiating about their requirements. In many cases, the participants’ Arguments led to a new Position or even further Issues/Sub-Issues. In the following Issue, one participant took an active role in discussing and proposing solutions:

**Specific Issue H** What comes first: Creating argument about a new model (or changes to a model) or creating a model (making changes) and recording arguments to support the model (the changes)?

**Context** We are discussing scenarios for creating/modifying a requirements model and documenting arguments for it. There may be two situations. In the first situation is a person can create a new version of the specification and may or may not record arguments for it. In the second situation, the person can only create/modify a requirements model based on an outcome of an existing argument.

... 

-P 1 Participant 2: Arguments come first

**AS** Every transition will be explained by at least one argument.

**AO** Participant 2: When you create a new model, you must have a transition. If you wish, you can have arguments linked to the transition to explain the reason for creating the new model. The model and transition come at the same time. The arguments can also be added later if you wish. It would be possible to
force the addition of at least one argument but I don’t think that is a good idea. It is possible to create a new model and not having any arguments.

**P 2** Participant 2:

Model: The new version of the model is created and a transition is created simultaneously to link it to its predecessor.

Argument: An Issue may be created at any time and linked to a model version. Positions can be added later as needed. An Issue without Positions or Arguments indicates to the designers that they may not have completed their task.

It is also obvious that the selection of Position cannot be based on simply the number of its supporting Arguments. Indeed, it depends on the trade-off analysis and negotiation about requirements between participants, for examples “I don’t think you should permit this to happen”, “Why is this only for creating arguments? This seems to me to be an edit function?”, “Can’t you simply have an argument linked to a model?”... In the above QOC analysis (see Figure 5.7), the participants set the priority for the Criteria. Therefore, the design explanation encouraged the participants to contribute to the requirements engineering task. However, not every Position was provided with both Object-Z and graphical explanation. In fact, the “formal” explanation was created only for complicated Positions and Issues.

The active contribution of non-specifier participants has also been observed by Johnson (1996). In his literate specification approach to using design explanation in supporting formal specifications (see Figure 5.8), non-specifier participants make contribution to formal specifications through reading and creating design explanation. The specifiers, however, do not create design explanation documents. Therefore, design explanation may be seen as a translator of formal language into the vernacular language.
However, the researcher’s experience of using design explanation herself when specifying the requirements and working with other participants led to a revision of the above model. Figure 5.9 shows the usefulness of design explanation in providing a unified view of different view on requirements and in supporting collaborative activities of various participants including the specifier in requirements engineering.

Reducing search time in the IBIS base

The IBIS base was found to be useful in providing essential input for the restructuring of the model. However, as identified from the previous research cycle extracting relevant IBIS arguments from the IBIS base was problematic (see Chapter 4). Indeed, searching for every issue related to the structure of specific classes and issues about their relationships required us to read in sequence the large number of all IBIS issues.
previously created and recorded in the IBIS base. Fortunately, in this research cycle, we did not have to access and read every issue previously created and recorded in the IBIS base in order to search for issues related to a specific problem. Instead, we read only QOC analyses (for example Figure 5.7) and only the issues which were created after that QOC analysis was structured. Let us examine the following selected examples\(^1\).

**Example 1** As discussed above, the QOC analysis (see Figure 5.7) was structured to review a number of IBIS issues related to the problem of re-defining a FOOM specification. This QOC analysis reexamines and summarises the issues, provides a holistic description of the situation and thus assists the specifier in solving to problem. This confirms a previous finding by MacLean and McKerlie (1995) that QOC is useful in summarising and generalising key attributes from various concrete, informal design representations. In this research, this usefulness was examined in more detail: can and how do we reuse the QOC summary? Later in our project, when solving a number of other problems of recording changes to a model and of writing scenario for creating arguments, the participants read this summarising QOC analysis among other QOC analyses and newly created issues. The QOC analyses saved them from reading all existing issues, \(i.e.\) they allowed them to work at a higher level of abstraction.

**Example 2** When specifying requirements which would enable the creation of argumentation documents in the form of node-and-link diagrams according to a specific argumentation notation, the specifier and her participants created a number of IBIS issues. At first, these IBIS issues were structured around a number of obvious classes, such as *Node*, *Link*, *Notation* and *Argument*. As they contin-

---

\(^1\)These examples are not necessarily given in the chronological order. They are given in an order which is believed to best explain my observation and analyse the findings.
ued to explore the requirements problem, they discussed also about other classes, such as *Project* and *Model*. The number of *Issues* increased overtime. Later, as a result of an insight (this will be discussed in Chapter 6), a QOC analysis was structured to review the IBIS issues in the light of the insight. The QOC analysis (to be presented in Chapter 6) provided a holistic view on the problem and design space around different classes (*Project*, *Model* and *Argument*). The requirements problem was ‘solved’. Later, the problem of the graph structure of the classes *Project*, *Model*, and *Argument* returned when the specifier and other participants had to specify different types of nodes of these graph-structured classes. They simply read through several QOC analyses (from which this QOC analysis was picked up for its relevance and was read in details) and the newly created *Issues*. As a result, another QOC analysis was created. Although these QOC analyses were created in order to take advantage of insight rather than to summarise IBIS *Issues* systematically, they also saved us time in searching the IBIS base later.

**Example 3** At a stage of development, the specifier had to solve a difficult problem of representing different types of FOOM models. As a result of an insight which happened when the specifier was exploring the problem, a QOC analysis was structured to reconsider the insight and reexamine the *Issues* related to the problem. Later, she decided to present the problem and her situation to a group of object-oriented requirements engineers/researchers to discuss the insight. First, she explained some *Issues*, then the specifier/researcher presented the QOC analysis for these *Issues*. The problem was quickly understood and discussed although there were other requirements engineers who did not directly involve in the project. With regard to the use of QOC, a general feedback was
that the QOC analysis was useful and enabled various (direct and non-direct) participants to understand the development stage and situation without reading in details specific issues.

The above example shows that by QOC analyses provided us with a summary of related IBIS issues at different development stages, therefore, saved us from reading the issues discussed when exploring specific problems. The last example also indicates that QOC analyses structured at different development stages could be used to communicate to other (and perhaps new) developers about the project. This led the researcher to the following examination of the use of QOC in overcoming the problem of accessing and searching in the large IBIS base.

![Diagram of Using IBIS within requirements engineering](image1)

![Diagram of Using QOC to supplement IBIS within requirements engineering](image2)

Figure 5.10: Using IBIS with and without QOC supplement within requirements engineering—the process

The advantage of QOC in reducing searching time can be qualitatively explained as follows. As the time progressed, requirements were explored, discussed and structured,
and our discussions were captured and documented using the IBIS notation. They were attached to the associated versions of the model. Figure 5.10(a) illustrates the previous research cycle, in which QOC was not used. In this research cycle, IBIS Issues were converted into a QOC analysis for the review of modelling options from time to time (see Figure 5.10(b)).

Over time, the IBIS arguments were accumulated and formed a linear stream—an unordered array of Issues. Although IBIS arguments were added to the base according to the time they were created, there was no better index than time to sort the Issues. A desirable index would be problem-based, but this is difficult:

- It is hard to classify/categorise the Issues. As discussed in Chapter 2, requirements problems are open-ended and ill-structured and may accommodate different interpretations. There might be many schemes to catalogue the IBIS Issues including those recorded and to-be-recorded. In addition, even within a catalogueing scheme, there is no clear cut between categories/subcategories of Issues: they might be named differently or overlap.

- An early decision about a catalogueing scheme for the organisation of the IBIS base might lock the requirements engineer into set categories of problems/sub-problems, ‘force’ her to see the requirements problem according to this decom-position scheme, and therefore it would not support creativity.

Therefore, the IBIS base was indexed, as is usual, according to the time Issues were recorded. As the model grows, the IBIS base becomes larger and more disorganised. Let us assume we have accessed the IBIS base in order to search and retrieve information from the base $k$ times. If $n_i$ is the number of Issues created during the $i^{th}$ period, there
are two possible situations:

**QOC is not used** If QOC is not used as in the situation of Figure 5.10(a) then the searching time in the IBIS base could be illustrated as in Figure 5.11(a). As the IBIS base is a disorganised array of issues, the only searching method applicable is reading all the issues in sequence, one-by-one. The lower bound searching time\(^2\) in the IBIS base would be:

\[
T_{IBIS\text{only}} = \sum_{i=1}^{k} n_i \tag{5.1}
\]

The reason we call this as a lower bound rather than an actual searching time is that an issue retrieved from the IBIS base may lead to a new concept, this in turn would require us to research for other issues related to this new concept. As the number of IBIS arguments increases, the likelihood of researching also increases.

**QOC is used** If QOC is used as in the situation of Figure 5.10(b) then the searching time in the IBIS base could be illustrated as in Figure 5.11(b). Searching and retrieving desirable information requires reading all previous QOC analyses and the last IBIS issues created during the \(k^{th}\) period:

\[
T_{IBIS\text{andQOC}} = N_{QOC} + n_k \tag{5.2}
\]

where \(N_{QOC}\) is the number of QOC created previously. It is an actual value

\(^2\)In these formulae, time is not measured in general terms, such as seconds or minutes, it refers to the number of times we need to access the design explanation base.
(searching time) while \( n_k \) is a lower bound. Indeed, each QOC document is a logical analysis of the global design space. Each QOC analysis restructures, reexamines and holds a number of context-specific related IBIS arguments together. Therefore, searching in these IBIS arguments again is redundant and going backwards. In most cases, \( N_{QOC} \) is expected to be \((k - 1)\), as a QOC analysis is expected to be created for each period. Hence, in most cases:

\[
T_{IBISandQOC} = (k - 1) + n_k
\]  

\( (5.3) \)

![Figure 5.11: Using IBIS with and without QOC supplement within requirements engineering—the searching time](image)

(a) QOC is not used  
(b) QOC is used

Therefore, QOC analyses reorganise the IBIS base and reduce searching time in the base.
5.3 How to use IBIS and QOC within FOOM effectively

5.3.1 A summary of findings

The analysis of observation data was also focused on how best to organise design explanation information, particularly how to use both *ad hoc* and *post hoc* design explanation techniques and notations effectively. Overall, learning from this research strongly confirms most findings from the previous research cycle. Furthermore, analysing design explanation information assists us in understanding FOOM modelling activities and reflecting on the IBIS and QOC notations.

With regard to understanding the process of documenting FOOM requirements evolution using both *ad hoc* and *post hoc* techniques, the learning:

- leads to a process model of structuring the graph of requirements evolution and documenting the rationale for the evolution
- confirms the classification of design decisions from the previous research
- suggests recording only arguments and the current version of the requirements model but not recording intermediate versions

With regard to the reflection of QOC and IBIS notations, the evidence:

- exhibits a limitation of the QOC notation in weighting Criteria
- confirms the usefulness of the added element Context and generates a current problem with using it
- encourages a flexible use of the IBIS notation
The analysis also leads to a comparison of IBIS and QOC arguments.

5.3.2 An analysis of findings

Documenting FOOM requirements evolution

The process of documenting the evolution of FOOM requirements using IBIS and QOC was analysed based on:

- the attachment of design explanation information to the requirements evolution (see Figure 4.6)
- the two types of arguments: Documentation and Transition
- issues identified in the previous research cycle. The issues were: when to release a new version and how to evaluate a candidate version versus existing version (see Section 4.3.2.

The analysis leads to a deeper understanding of the FOOM modelling process at a micro-level (see Figure 5.12). The FOOM modelling process can be seen as rather iterative. Each iteration consists of discussions on a particular requirements problem generated from a selected version and evaluation of possible changes to the version to solve the problem. When a version is selected for discussion, for example the version MAW7.2 in the previous section, a temporary version is created as a candidate new version. The selected version is fixed while the candidate version is subject to intensive evolution moving towards a relatively stable and satisfactory stage. Changes to the candidate version and the underlying deliberation are documented using IBIS, for example the three IBIS arguments: Issues E, F and G in the previous section. When
it comes to making a *final* decision on the problem, a *post hoc* review may be conducted using QOC, for example the QOC argument presented in the previous section (see Figure 5.7). Therefore, all possible alternative models are gathered and examined thoroughly. If the QOC analysis confirms the “original” selected version then there is no new version to be released, and the candidate version is removed. Arguments which support the appropriateness of the selected model are attached to the *Documentation* of the model. However, if the QOC review supports the candidate version, then the temporary candidate version is released as a newly created version. Arguments which support changes to the selected model are accumulated in the relevant *Transition*. In the example discussed in the previous section, a new version was created. The previous QOC argument was therefore attached to the *Transition* from MAW7.2 to the new version MAW 8.0.

Two further issues generated from this process model are whether our classification of arguments is appropriate and whether all temporary states of requirements models should be recorded.

This classification of arguments into the two types of *Documentation* and *Transition*, proposed during the previous research, is questioned for two reasons.

First, one may argue for another classification of arguments that all arguments support either the currently selected model or a new model, if there is one. For example, if an argument supports a transition then it can be attached to the newly created version. This classification is possible. However, we would prefer the classification of arguments into *Transition* and *Documentation* for it expresses a meaningful dynamic process of a requirements evolution, not merely the static states.
Second, there are many open Issues. They are Issues without any Position at the time the Issues are recorded, or with only Positions for further consideration (?P) and/or rejected Positions (-P) but without any selected Positions (*P). Issues without any Positions or Arguments indicate to specifiers that they may not have completed their task. These issues may be returned to and resolved at a later time. There is a question about whether to keep and how to classify these issues. At present, these issues are attached to the Documentation of the Model under discussion since they have not resulted in any changes. Issue A is an example of an unsolved issue. The absence of a selected Position
in an open issue indicates that the issue requires further attention.

Therefore, all design explanation arguments are attached to either a model they support or a transition they argue for.

Recording all temporal specifications of all FOOM components as a FOOM specification evolves is too expensive and complex. There is a complicated problem of granularity ranging from a more or less complete FOOM specification, a diagram or a “full” Object-Z specification and to a FOOM node or link (Object/Class) in the diagram or each Object-Z class. Furthermore, evidence shows that the explanation information about a particular (most often a current) FOOM specification is more often desirable than the reconstruction of all its temporary states. In addition, if design explanation is well documented and organised, then temporal specifications can always be reconstructed when needed.

**Reflecting upon the IBIS and QOC notations**

The QOC notation gives the specifier and other participants an expressive and explicit representation of a logical space around a specification with possible options assessed by a number of criteria. The assessment of options against a common set of criteria promotes a comprehensive evaluation of different specification options. However, this leads to a problem of weighting criteria. A question is not solved merely because of the number of supporting criteria to options. Sometimes a criterion which is the most important may override all the other criteria. At present, we simply organise criteria in the order of their importance/priority, the most important is written first and less important criteria are written below (see Figure 5.7). This issue has been identified in the literature. Shum (1993) suggested using criteria trees to represent the
relationship between Criteria ranging from general criteria to more specific criteria in a design context. Khushalani (1997) (a non-QOC developer) suggests using number to prioritise Criteria. However, there has not been further study into the evaluation of these techniques (see also Chapter 2). In our experience, the simple representation of Criteria in the priority order helped us avoid extra effort of the participants. The weighting of Criteria was done informally through discussing and negotiating requirements between various the participants. Clearly, the issue of weighting Criteria needs further research attention.

The evidence also shows that the IBIS notation and the newly added element Context are very useful to capture on-the-fly FOOM modelling discussions. The above IBIS examples demonstrate how Context explains why an Issue is initiated or explains the problem we wanted to solve in more detail. Context is also highly desirable for the assessment of how an Argument supports or objects to a Position, especially when the same Position may be assessed differently in different Contexts. For example, various Positions of Issue F and G were examined from different perspectives, thus were solved differently. The usefulness of Context can be examined in the light of the evolution (and volatility) of requirements as follows.

During the requirements engineering process, the specifier and her participants moved around the problem space. From time to time, we came back to reconsider or re-solve a previous Issue after exploring other Issues. Every time we revisited a previous a problem (Issue), the problem situation might be different. Let us consider an evolution of the class Model. Initially, the class Model was defined as a set of design components—instances of the class Component. Later, as a result of the restructuring the graph structure of the classes Project, Model and Argument, Model was redefined
as a graph—a concrete subclass of the class VirtualGraph. When specifying the class Project, the specifier viewed Model also as a Node. This lead to the removal of the class Component. Model was considered as a graph of Node if viewing it from the local structure perspective, but it was also considered a subclass of Node if viewing it from the Project level. After the introduction of the class Specification, Model was no longer a Node. The definition of Model evolved as the requirements problem was continually explored and considered from different angles. So did the status of Positions and associated Arguments about the class Model and its relationships to other classes. Later, the class Model was specialised into different subclasses of node-link diagrams and ‘pure’ graphs. The element Context was very useful in two ways. First, it provides a background—problem situation for our current debate and discussion about requirements. Second, it refreshes the specifier’s and participants’ memory and explain why a certain issue was solved differently in the past or why we should reconsider the issue. This is especially important as sometimes a decision might have ‘ripple effects’ on previous decisions. A further question is: How can we trace the evolution of Context so that we can be aware of what issues may need to reconsider when new Contexts take place? A possible solution may be to formalise the relationships between the added element Context to other IBIS elements needs. However, the formalisation of Context (such as using temporal logic (Ramesh and Dhar, 1992)) may decrease the attractiveness of IBIS which lies on its simplicity. Notational issues were however not my primary research objective, I would prefer to investigate this issue in a future project.

We have also experienced a rather flexible use of IBIS. This is consistent with the findings of Potts and Takahashi (1993). Issues may be recorded incompletely without Positions, or Positions may be recorded without Arguments, if Positions/Arguments are...
not available. Some Positions can be rejected or selected quickly without recording an Argument which is too obvious. According to IBIS, only an Issue may generate or specialise a sub-issue, however, sometimes it is better to record that a Position may lead to a sub-issue (Issue).

**Specific Issue D** Should we allow create or document arguments?

**Context** Creating an argument means doing rationale, i.e. creating Issue, looking for Positions, adding Arguments, only then deciding which Position to select. The resolution of the argument will decide whether of Documentation and Transition will keep the Argument. On the other hand, documenting an argument means recording a more or less complete Argument. This issue was created first by the specifier when specifying the operation CreateArgument of the classes Documentation and Transition. The issue was then discussed with Participant 2.

?P 1 Allow both to create arguments as well as to document complete arguments.

**Sub Issue A2-I1.1** Must Arguments be grouped in a Documentation or Transition?

*P Yes.

AS I don’t think we should permit any Argument not being grouped in any Documentation or Transition. (Participant 2)

...

?P 2 Allow only to document arguments.

...

It is evident that the Sub-Issue is meaningful and represents an actual problem only within the Position P1. If a Position is discarded, then the associated Sub-Issues will not be active. This suggests an extended link between Position and Issue for the current IBIS notation.
Approach

IBIS: \textit{ad hoc}: IBIS is used to record on-going incidents and activities (FOOM process).

QOC: \textit{post hoc}: QOC is used to build a logical map to the evaluation of FOOM specifications, not the history of FOOM process.

Elements

IBIS: Issue, Context, Position and Argument.

QOC: Question, Option, Criterion, and Assessment Argument.

Outcome

IBIS: A on-fly record of research process.

QOC: A reflection at a higher level of abstraction by the specifier.

Strengths

IBIS:
- Good for quick note-taking during discussions and planning meetings (without evaluation criteria),
- Can be used non-intrusively to FOOM modelling,
- Provides a chronological and detailed record of activities,
- Provides a useful resource for constructing QOC.

QOC:
- Has a set of common and explicit criteria for evaluation of different Options,
- Encourages the generation of evaluating Criteria,
- Provides a broader view and a logical map to a requirements problem,
- Can be used as a review of IBIS information.

Weaknesses

IBIS:
- Implicit evaluation,
- Locality of issues.

QOC:
- Documentation of QOC requires additional separate time from design activities.

Table 5.1: Differences between IBIS and QOC collected data

Comparing IBIS and QOC information

The evidence also brings forward a comparison between IBIS and QOC information created within one project. This is not a new debate on the differences between the two design explanation approaches. However, this is the first time that the two notations are used in a complementary way in one requirements engineering process. Table 5.1 represents our comparison of IBIS and QOC arguments recorded during this research.

Both IBIS and QOC information was very useful to the FOOM process including elic-
itation, modelling and evaluation of requirements (see Figure 5.3). During modelling, IBIS is non-intrusively used by the specifier to record her own reasoning and discussions with other participants. The IBIS records serve as more than just well structured *ad hoc* design memoirs, they also encourage accurate requirements modelling. Therefore, we believe that using IBIS improves the quality of the FOOM process.

Nevertheless, while IBIS provides an excellent mechanism to capture on-the-fly process of FOOM specification, it is hard to construct a broad logical analysis and evaluate alternative models (or parts of a model) documented in different local IBIS issues. QOC arguments were very useful in restructuring and re-examining IBIS arguments using a *post hoc* problem solving approach. The next paragraph explains how QOC arguments can be constructed from related IBIS arguments.

Each **Question** represents a common requirements problem among the related issues. Each **Option** represents a modelling option, taken from **Positions** in IBIS arguments. Each **Option** often represents an existing or a candidate version. Some **Positions** may be expressed in different **Issues** but they are represented as one **Option** if they describe the same specification for the requirements problem (**Question**). In contrast to **Positions** in IBIS, **Options** are assessed against a set of common and more explicit evaluation **Criteria**. The QOC notation encouraged us to generate **Criteria**. **Criteria** can be generated from a global view of clients’ requirements. **Criteria** can also be found from IBIS Arguments but are generally more abstract than concrete Arguments. While (IBIS) Arguments often are expressed differently, for example “this information could be deduced from...”, “this is derivable information”, “there must be an explicit link”, they all can be expressed by one **Criterion**—expressiveness of the model. IBIS Arguments can serve as the **Assessment Argument** in an extended QOC (MacLean et al., 1991b) (see Figure 5.7 for illustration).
The automation converting from a number of IBIS arguments to a QOC argument would be desirable.

Is there overlap between the two types of information? At the present, the answer is that there is overlap. However, as demonstrated in the examples discussed above, QOC arguments do not provide a full documentation of the modelling process while IBIS arguments document one issue after another as they occur. In particular, QOC arguments provide logical maps around FOOM specifications and explain how different IBIS arguments hold together.

5.4 Conclusion

In conclusion, it is clear that the supplementation of the *ad hoc* IBIS arguments with *post hoc* rationale constructed using QOC offered the FOOM specifier many advantages. The application of both notations demonstrated that both notations were useful and desirable for the specifier and her participants. The research cycle also resulted in a deeper understanding of the process of documenting the evolution of the requirements specifications and the rationale for the evolution. In addition, the findings lead to a few suggestions about how to use both the QOC and IBIS notations effectively.

Finally, let us come back to early issues highlighted in the previous section: the cost of the creation of QOC and when best to QOC to reduce the extra effort of creating it? As shown in this chapter, there are benefits of using QOC to supplement IBIS. However, they come at a cost. The real issue should be: when (and how) to use QOC to most effectively leverage its benefits? This is also a subject of current debate in research into design rationale (see Chapter 2). The use of design explanation should be
understood and addressed in the context of the process it tries to improve. The new understanding of the requirements engineering process briefly described in Chapter 4 seems to hold the key to address the issue. This will be analysed in Chapter 6.
Chapter 6

New Understandings of the Requirements Engineering Process

In Chapters 4 and 5, I described the first and second research cycles which explored the use of *ad hoc* and *post hoc* design explanation within the requirements engineering process. Having reflected upon their application, I was able to conclude that IBIS has both benefits and limitations and that QOC is useful in overcoming the IBIS limitations through:

- reorganising and inspecting the IBIS base,
- providing a holistic view of the problem situation, and
- reducing the search time in the design explanation base.
Therefore, the use of *ad hoc* and *post hoc* approaches to design explanation in parallel has been confirmed to be both complementary and useful in requirements engineering.

Two interesting issues arising from this investigation include: “when is it appropriate to do QOC?” and “is the cost of the creation of QOC design explanation justified?”

As discussed throughout earlier chapters, the primary objective of this thesis is to investigate the feasibility of using design explanation within requirements engineering in order to develop a new approach to monitoring the requirements process and improving understandability of the requirements specification. A quantitative cost benefit analysis of the approach is not appropriate at this stage but will be required once the approach has been conceptually and qualitatively proved feasible. Appropriate questions at the present are: “when to do QOC to leverage its benefits?” and “how to use IBIS and QOC in parallel effectively?” These questions are not totally new. As the discussion in Chapter 2 illustrates, these questions have been concerns of the design explanation research community for years.

There have been attempts by the QOC community to address the question of when to create QOC. At present, there is a general agreement that QOC should not produce a complete description of the design space; it should be created to solve critical problems (see Chapter 2, Pages 81 and 87 for details). However, in previous research, the QOC approach has been used in isolation from all other design explanation approaches. Our investigation, thus differs from previous research and is characterised by the following features:

- QOC is used to supplement IBIS.

- Both the IBIS and QOC bases serve as data for the researcher to examine and
understand the requirements process.

- The context when a QOC analysis is triggered is examined, in detail, from the requirements engineering process perspective.

- The process of using both IBIS and QOC within FOOM is continually observed, reflected upon and revised as a result of hermeneutic cycles of an action research study.

Therefore, the above questions led to a post hoc examination of the requirements engineering process in detail and the application of both the IBIS and QOC notations from the process perspective. The chapter is structured as follows:

**The catastrophe-cycle requirements modelling process** This section presents the researcher’s overall interpretation of the requirements engineering process. An overall pattern emerges.

**Complexity of the requirements model** This section analyses the process in more detail through two selected examples and identifies two types of complexity—essential and incidental. A more detail pattern is presented.

**Supplementing IBIS with QOC—Increasing essential knowledge** Based on understanding gained from the previous sections, this section analyses in details what is happening at critical points.

**Summary** The section summarises the new understanding gained from the above analyses.
6.1 The catastrophe-cycle requirements modelling process

This section presents a qualitative interpretation by the researcher of the requirements engineering process. It is based on the observation of the process and qualitative data recorded during the two research cycles (see sections 4.1.3 and 5.1.3 for details). Overall, the interpretation shows that the process of requirements engineering is creative and rather opportunistic. Particularly, it also leads to a new pattern describing the evolution of the requirements model.

The FOOM process was confirmed to be cyclic, each cycle consisting of elicitation (E), modelling (M) and evaluation—validation (V) activities. According to the observation, the process consists of many fast cycles rather than few extensive cycles. The data show frequent movements forward and backward between acquiring and understanding information from problem domains and representing and validating it in the requirements model. This process reflects a normal requirements engineering practice and also confirms the description of these cognitive behaviours offered in the literature (see section 2.1.2).

Initially, the complexity of the requirements model was expected to increase incrementally over time as illustrated in Figure 6.1. The requirements engineering process was expected to be an incremental evolution during which the incompleteness and ambiguity of the problem is gradually reduced as implied in the literature (Malhotra et al., 1980; Christel and Kang, 1992; Simsion, 1994; Loucopoulos and Karakostas, 1995). As a result, the requirements model would grow more complex and complete as more and more objects/classes are identified and their specifications are refined.

One unexpected result of this research was that the evolution of complexity of the
FOOM model was not in accordance with the above expectation (see Figure 6.1) but rather exhibited occasional, but major, restructuring (see Figure 6.2). This could be explained by the inevitable increasing entropy during the modelling process. There were a number of crisis points during the process. These crisis points occurred when requirements addition/modification suggested that the model should not evolve further without being restructured.

Indeed, the history of the design process documented using IBIS and the intermediate
versions of the requirements model shows an increase in knowledge and a growth in complexity over time. Some intermediate versions, and changes to them, show that there were occasional simplifications and major restructuring of the requirements model. At these moments, the complexity of the model was reduced significantly\(^1\). These simplification/major restructuring points are, in fact, critical points at which problems are reconceptualised and solved, thus they mark the end of cycles of the catastrophic accumulation of the complexity. The pattern, therefore, might be considered as a catastrophe-cycle requirements modelling process\(^2\). The oscillations of the complexity of the requirements model can be explained as follows:

- As time progressed, the designers’ knowledge of the problem space increased, the problem was explored and structured, i.e. modelled. Indeed, classes in object-oriented diagrams, their properties and relationships and communication between them were incrementally introduced and modified as new information was acquired, clarified, modelled and validated. In this way, the understanding of the problem was gained and developed. This is consistent with Visser’s (1992) argument that, with ill-defined problem situations, designers construct the problem space and with Guindon (1989) who also describes design as a knowledge discovery process.

- As understanding of the problem matured, partial problems were explored, solved and, in turn, triggered further problems. Working on a requirements problem and/or a group of classes led, from time to time, to the unplanned creation of

---

1. The thesis does not attempt to give quantitative measurement of the complexity over time because of the qualitative nature of data and the researcher’s desire to achieve a deep contextual understanding of the process.

2. Here, the word “catastrophe” is used with its conventional English meaning, it is not used as a technical term in the sense of, e.g. catastrophe theory.
new classes or the modification of existing classes. The IBIS base shows that the path that led from one class (subproblem) to another was rather unpredictable, but clearly different from the systematic decomposition approaches described in the literature (Kant and Newell, 1984; Jeffries et al., 1981). Often in our studies, problems exposed and documented in Arguments supporting or objecting to possible solutions for an Issue, led to either previously discussed or newly generated Issues/Subissues. Existing Issues were sometimes reviewed and resolved and subsequent Issues were often discovered and created (for examples see numerous Issues presented in Chapters 4 and 5). The path between Issues was drawn as the problem space was uncovered and explored. Finding a Position and/or choosing one among a number of Positions for an Issue might uncover an unexpected view on the problem and lead to an unexpected path to solve it. The movement from Issue to Issue exhibited ad hoc and opportunistic process characteristics, not those of a strictly and systematically predefined plan (see a discussion about a linear stream of IBIS Issues in Chapter 5, Page 202). This is consistent with Schön’s (1996) description of professional practice: “As you work on a problem you are continually in the process of developing a path into it” (Schön, 1996, page 175).

All the participants (including myself) in this requirements engineering project agreed that raising “appropriate” Issues and exposing solutions and Arguments relied greatly on the participants’ personal experience and creativity.

- As a result of the exploration and modelling of the problem space, the complexity of the requirements model progressively increased. This is illustrated in Figure 6.2 by the rising curves. The more complex the model became, the harder it became to add and fit new components to the growing model. The complexity
grew rapidly. A more detailed analysis will be provided in the next section.

- At some stage, the model was reconceptualised: it was simplified and restructured. Thus the complexity of the model, considered as for example a digraph or a compound logic statement, significantly dropped. This is illustrated by the falling line in the diagram. Often, the simplification was driven by insight rather than systematically deliberate efforts. It is very important to note that as a result of insight, a new way of understanding and conceptualising the problem suddenly became apparent. Consequently, the requirements model was changed substantially in that an alternative architecture of the model was derived. The simplification of the model should not be understood merely in terms of the reduction of a number of components of the model.

- After reconceptualisation, the newly simplified model became the basis for further development cycles.

Although the catastrophe-cycle modelling process is revealed through examining the IBIS base, this does not appear to be caused by the use of IBIS, but rather to be thrown into sharp relief by the research focus. This is confirmed by a post hoc qualitative examination (Nguyen et al., 2000) on the process of problem understanding of other field studies (Carroll and Swatman, 1999a; Carroll and Swatman, 1999b). These studies were also longitudinal and involved commercial projects, one in a small consulting company, the other in a large manufacturing company. Although these field studies involved a variety of requirements engineering methods and did not involve any design explanation approach, an analysis at a high abstraction level also shows similar results.

The requirements engineering process is described as creative and involving opportunis-
tic exploration and adaptive responses to the problem state. The analysis path is rather unpredictable, neither top-down nor systematic. The requirements engineer often explores one problem area, gathers some information, moves on to other areas without even tentatively solving the problem, and later comes back to reconsider or resolve the previous problem. As a result, “understanding and complexity are built over time” (Carroll and Swatman, 1999b, page 5). Periodically, the requirements engineer has an insight into understanding the problem and reconceptualises the problem situation and restructures the information acquired, consequently representational complexity is reduced (Carroll and Swatman, 1999b).

In summary, we view RE as a creative process that involves oscillations of complexity, described by the catastrophe-cycle model. Each cycle consists of constructing the problem space, developing a mature understanding of it, structuring a model to represent the problem, inadvertently building up the complexity of the model, and then significantly simplifying the model by insight or critical thought. The restructuring is insight-driven rather than based on systematic efforts. As a result of insight, the restructured model has a new architecture representing a new way of perceiving and understanding the requirements problem. The reconceptualised problem is then further developed in the next cycle.

There is another important observation: although, throughout the process, the inherent understanding can only increase—not reduce, the overall complexity of the model was reduced occasionally! This led the researcher into a more detailed analysis of the complexity of the requirements model and its dynamics. The next section identifies the essential and accidental complexity of the requirements model, investigates their evolution and refines the catastrophe-cycle model of the requirements engineering process.
6.2 Complexity of the requirements model

The post hoc analysis of the complexity of the requirements model in the two research cycles identifies two types of complexity: the essential and the incidental. The former represents the inherent understanding of the problem space displayed in the requirements model and the latter represents the complexity of expression and/or structure rather than substance in the model. The observation of the dynamics of the complexity was consistent throughout the process and could be illustrated with examples extracted from the recorded IBIS and QOC base. Let us examine the complexity of the requirements model through the two following examples which were selected for the sake of brevity, ease of understanding but most importantly for their consistency with the other examples. The first example will be analysed in more detail in terms of issues and intermediate models recorded, and the second example will be analysed in more detail in terms of the cognitive aspects: analysis path, breakdown and insight.

6.2.1 Building up an initial understanding of the complexity of the requirements model

The first example

Here, I analyse the first example from Chapter 4 in the light of the above discussion. As described in Chapter 4, one of our tasks was to specify requirements for a CASE tool which supports the creation of argumentation documents in form of node-and-link diagrams according to certain rules (of a specific argumentation notation). The argumentation documents created when the CASE tool is eventually used should be linked to the associated requirements model.
Initially, the specifier/researcher and her participants acquired information and developed and refined our understanding of the problem. We identified obvious classes, such as Node, Link, Notation and Argument, drew a sketch of the Object-Class (O/C) structure diagram and discussed the relationships between these classes (see Figure 6.3(a)). In this early model, each Argument is an aggregation of objects Node and Link (see Issue I, Position 1). Each object Notation defines a set of rules—“legal” types of Node and Link and ways in which they may be connected. Later, we saw that Link may be considered to be a subclass of Node, with the property that instances of this class could link two other objects of (subclass of) class Node. This is reflected in Position 2 of Issue I which discusses how to specify the class Argument. The IBIS arguments show clearly that, as time progresses, further details of the problem are uncovered and modelled. The growth of our understanding of the problem led to modifications/changes to requirements which increased the complexity of the model.

**Specific Issue I** How to specify the class Argument

**Context** The class Argument represents an argumentation document. It is a node-and-link diagram. In a previous version, this class was seen as an aggregation of instances of the classes Node and Link. However, now Link is considered as a subclass of Node, the definition of Argument is revised.

*P 1* Keep the current version

```
Argument

nodes : P Node©
links : P Link©
...
```

AS It represents expressively the node-and-link structure of the rationale document
Argument is an aggregation of instances of the class Node and its subclasses.

\[
\begin{align*}
\text{Argument} & \\
\text{nodes} : & \mathcal{P} \downarrow \text{Node}\circ & \\
\text{...} & & 
\end{align*}
\]

**AS** It elegantly expresses the node-and-link structure of the rationale document and the hierarchy between nodes and links.

As the specifier continued to elicit and analyse requirements, she added more and more features to existing classes and created additional classes (Project and Model) and their relationships. The inherent understanding of the problem increased progressively through activities of further elicitation and refinement of requirements. As the focus of our activities switched towards the representation and the evaluation of the solutions, the requirements model became more complex (for example see Figure 6.3(b)).

Following this, new components were added, specifically the classes Project, Model, Object and Decision and their relationships. The class Project represents the evolution of the requirements model. The class Model represents intermediate versions of the requirements model (see Figure 6.3(b)). **Issue J** defines the class Project as an aggregation of instances of the class Model.

**Specific Issue J** How to specify the relationship between Project and Model

**Context - Position 1** Association
Figure 6.3: Evolution of requirements for the CASE tool (extracted from intermediate specifications)

\[
\text{Project} \\
\text{versions : } \mathbb{P} \text{Model} \\
\ldots \\
\forall \text{model : versions} \cdot \text{model.project} = \text{self} \\
\ldots
\]

\textbf{AS} Project keeps references to instances of Model not actual copies of every versions of the specification.
This is rather an implementation matter.

**Position 2** Aggregation

\[
\begin{array}{l}
\text{Project} \\
\quad \text{versions} : \mathbb{P} \text{Model}
\end{array}
\]

\[
\begin{array}{l}
\quad \ldots \\
\quad \forall \text{model} : \text{versions} \bullet \text{model}.\text{project} = \text{self} \\
\quad \ldots
\end{array}
\]

There is not just a simple association, each Project aggregates different versions (instances of Model) of a specification. This supports a view that two versions may be identical but they are two instances of Model.

Clearly, our understanding of the problem increased as we continued carving an appropriate path into the problem. We added more components (the classes Object and Decision, their attributes and relationships) to the models (see Figure 6.3(c)). Object is defined as a design object (component) of a requirements model. Issue K defines the class Model as an aggregation of the class Object.

**Specific Issue K** How to define the structure of Model

**Context** We decided to attach Arguments to components of Model rather than just to Model. Therefore we created the class Object to represent design objects/components. This Issue discusses the structure of Model.

**Position 1** Model is an aggregation of design objects

**AS** For the time being, a model is a set of design objects

**AS** Arguments may be attached to design objects to trace the development of each design object

**AO** This specification does not include the connection between design objects.
Decision is the resolution of an Argument. Its purpose is to explain the relationship between Models involved in a change (a transition) and associated Argument (see Figure 6.3(c)). The evolution link between different versions (instances of Model) is discussed in Issue L—it is later specified in the class Decision.

Specific Issue L How to specify links between versions (Model) within a Project

Context Here we discuss how to record changes to the requirements model.

-Position 1 Project will hold a relation Model ↔ Model

AO How to extract Arguments associated with a particular Model (and changes to that Model)?

*Position 2 There should be another class (which could be named as Delta or Decision) which explains the relationship between Model involved in a change and associated Argument.

AS This class holds a set of Argument explaining a transition from a Model to another.

The Issues I, J, K, L together with other Issues recorded during the process show clearly that as time progressed further details of the problem were uncovered and structured. The model evolved through a number of changes and became more complex (see Figure 6.3(a), 6.3(b) and 6.3(c)). The growth of our understanding of the problem led to addition of and/or changes to requirements which increased the complexity of the model.

The above changes to the model had to fit into the existing model. Indeed, each of the changes instigates a number of IBIS arguments about the relationships and communication between the newly discovered classes and the existing structure diagram. Some important IBIS arguments were:
• How might we accommodate the class Decision within the relationship between Project, Model and Argument?

• How might we define the structure of Model?

• How might we specify the relationships between Object and Model/Argument?

Therefore, the complex structure of the growing model forced complex expression of newly discovered information (see Figure 6.3(c)). New information was added in an inefficient way, therefore exacerbating the complexity of the model. The more complex the model is, the more complex the expression of the new information becomes. The growth of incidental complexity (complexity of expression rather than substance) overwhelmed the growth of the essential complexity of the model. The overall complexity increased rapidly.

Next, an unexpected insight led to exciting progress. The abstract class Graph was created to generate the graph structure for the classes Project, Model and Argument. The models underwent a major restructure (see Figure 6.3(d)). The advantage of this restructure is described as follows.

In more mature professions, (e.g. construction engineering, medical science, legal decision making) practitioners tend to apply an established fundamental theory of their field to a concrete situation. In requirements engineering, however, there is no well defined theoretic toolkit by means of which we may transform the client’s incomplete and ambiguous requirements, especially in the case of a rather broad and undetermined problem space, into a formal or semi-formal specification. This activity often involves novel uses of techniques and approaches applied to unfamiliar systems and/or domains (Guindon, 1990b). This problem solving process requires creative activities and heuris-
tic methods rather than prescriptions. Not being armed with a well-established theory, requirements engineers must gather their (holistic) understanding of the case problem, classify objects into categories (they might have to form new concepts), relate the gained understanding to their repertoire and represent requirements using a modelling notation. The problem understanding and solving activity of requirements engineering, therefore, tends to be insight-driven. A poor classification can lead to poor quality models—sometimes to a blind alley, while an appropriate concept/insight can accelerate the development process and elevate the level of abstraction of the model.

In this case, the discovery of the graph structure of classes *Project*, *Model* and *Argument* served as a critical turning point. The establishment of these classes as subclasses of the newly created virtual class *Graph* (see Figure 6.3(d)) was significant:

**It deepened our understanding of the problem** Prior to this point we focused on local understanding around the structure of each class of the model. The new classification reflects a holistic view of the model.

**It accelerated the specification process** Our models are now based on well-established Graph Theory. Later, we apply the concepts of cyclic and acyclic structure of the Graph theory to specify our specific classes. This increases the extendibility of the model.

**It simplified the specification** Due to the specialisation of the virtual class *Graph*, the formal specification of classes *Model* and *Argument* became easier to write. Indeed, common properties of the classes *Node*, *Link*, *Notation* and *Notations* are lifted up to the virtual class *Graph*. This increases simplicity and extendibility of the model.
The overall complexity of requirements models was reduced significantly. However, our conceptual understanding of the problem must have been increased. Clearly, the complexity of the model cannot be seen merely as a function of its representation. In fact, this simplification of the model shows clearly the reduction of incidental complexity of the model resulted from an increase in (discovery of) new information/understanding of the problem. The inherent understanding gained and embedded in the model can be called the essential complexity of the model.

The second example

The next “cycle” (of building the complexity) started with a further specification of the graph structure for classes Model, Argument and Project. The IBIS documents show the specifier’s and her participants’ attention switching between different problem areas rather than working with each one in sequence. The specifier drifted between different ideas and issues. Both their understanding of the problem and the incidental complexity of the specification increased over time. There are important issues which “shape” the growth of the complexity of the requirements model. These issues are presented according to the time order as follows:

- There was a difference between the graph structures of these classes. In contrast to Argument and Model, which could allow cyclic links, the network of Models (versions) in Project was determined to be acyclic. The rationale supporting this determination was that we wanted to maintain consistency with the time factor: A version created later cannot lead back to another version created earlier. If this happens then a copy of the latter will be seen as a newly created version. Therefore, the class Project was discussed and further defined as a subclass of the
class *AcyclicGraph* (see Figure 6.4(a)). The recorded IBIS arguments and interim versions of the requirements model show the gradual growth of the complexity of the model.

![Diagram of AcyclicGraph](image)

Therefore, the participants’ attention was switched to the class *Decision* when specifying the graph-structured classes. The above issue led to a redefinition of the class *Decision*:

![Diagram of Decision](image)

*The class Decision records a change from a Model (a version) to another Model (another version) The change is supported by a set of arguments. In other words, this class records references to the evolution of models and the underlying rationale.*

![Formula](image)
This definition needs to be refined to support the acyclic structure of the Project graph, including the situation in which arguments and design decisions confirm the appropriateness of a Model—decisions not to change a current version. The following situation can be expressed as:

\[ \exists d : \text{Decision}, \ arg : \text{Argument}, \ m : \text{Model} \implies d = (a, (m, m)) \].

If this happens, a new instance of the class Model, \( m_{new} : \text{Model} \), would be created, so that values of the properties of \( m_{new} \) are equal to values of the properties of the existing \( m \) while \( m_{new} \neq m \). An operation was written specifically to do this for the class Model. However, an Object-Z facility for “cloning” instances in general would be desirable. There were also a number of issues related to the problem of how to record decisions and transitions between Models while maintaining an acyclic graph structure of Project: in the properties or operations of the classes Project and/or Decision. These issues show that the modelling activities were switched between different problem areas and at different levels of abstraction.

As a result, the definition of the class Decision was revised to support the acyclic structure of the class Project and while still fitting into the existing associations with the classes Model and Argument. The rate of growth of the overall complexity was slowing down. Later, this sub-problem was solved by the specification of the class Decision into the two subclasses Documentation and Transition (see Figure 6.4(a)). (See Chapter 5 for detailed specification of these classes)

- The focus then was switched back to the graph-based classes. The QOC analysis from the example 1 was retrieved to review these classes. Having this summary

\[ \text{This decision was revised later during the requirements engineering project, however, the revision is out of the scope of the current discussion about the second example} \]
in the form of QOC analysis assisted us in being focused on the current task rather than searching and reading all associated issues previously created. This use of QOC is analysed in Chapter 5. Although the classes Project, Model, and Argument share a common graph structure, they are composed of different types of node:

- Apart from being a graph of design objects (class Component/Object at this time) in its own right, a Model is also a snap-shot of a requirements specification (called a version) at a specific time.

- An Argument is a graph of Nodes and Links and is attached to a specific Model.

- A Project is a graph of Models.

• There were a number of IBIS issues related to this problem which prompted us to reconsider the requirements and the structure of the classes. The ramification of this problem turned out to be more important than expected. In fact, we learned that there is a recursive graph structure in the system being modelled. The system consists of different Projects, where each Project is a graph of nodes. Each node is a graph (Model), which in turn consists of nodes of other types (Node/Link). This graph (Model) is also attached to a number of other graphs (Argument) consisting of Node/Link. The unforeseen recursive graph structure of the classes uncovered a new understanding. This understanding exposed for us an unexpected complicated problem space. The model was no longer seen as consisting of separated graph-based classes, but a complex recursive graph of graphs of different types. The problem complexity was increasing rapidly.

We created the virtual class GraphNode and started building an inheritance tree
of different nodes for different graphs. The modelling subprocess became very complicated and error-prone. The overall complexity grew rapidly.

- It suddenly became apparent that Model could be a kind of Node. This insight resolved the problem by allowing the use of the multiple inheritance structure for the class Model instead of the complicated inheritance structure for the nodes. A QOC analysis was constructed to assess the new situation and help the specifier and her participants implement this insight (details will be provided later). The complexity of the specification was reduced significantly in the following ways:

  - The classes GraphNode and Component (see Figure 6.4(a)) were eliminated
  - Apart from being graph of Node and Link in its own right, Model is also a-kind-of Node
  - Graph is a network of all derived Nodes,
  - Project is a network of Models,
  - Argument is a network of Nodes and Links.

![Figure 6.4](image.png)

Figure 6.4: Improved classification as a result of insight (extracted from intermediate specifications)
This story shows how the overall complexity of the requirements model increased gradually as the problem space was explored, different ideas arrived and new understanding (and sub-problems) emerged. At some point a hidden problem suddenly becomes apparent: the recursive graph structure of the requirements was realised. This can be compared to Heidegger’s (1967) concept of breakdown: “objects and their properties are not inherent in the world, but arise only in an event of breaking down in which they become present-at-hand” (Winograd and Flores, 1986, page 36). To a problem solver, the course of action is interrupted by unreadiness when a hidden problem becomes apparent (Winograd and Flores, 1986) (see also Chapter 2). The complexity increased rapidly as the requirements engineer added new constructs to the requirements model in an attempt to solve the complex problem at the unreadiness point. In addition, she also tried to make new constructs fit with the existing structure of the model. Later, the problem was solved and the complexity was reduced significantly. This happened as a result of an insight which reconceptualized her understanding. This can also be compared to what is described in Mayer (1992): it is often that that the problem solver does not have accurate feeling of when he/she would solve the problem. Insight, a well recognised psychological phenomenon (see for example, Mayer, 1992), often involves surprise. In fact, the example above also evidences the insight-driven nature of requirements engineering.

**Discussion**

Before further discussing essential and incidental complexity in a requirements model (in the next subsection), let us look at a preliminary interpretation of the requirements evolution in the above examples.
These examples, in common with many others in the recorded IBIS and QOC base, depict the requirements evolutionary process as periods of incremental development interleaved with events of major restructure and simplification through reconceptualisation. Although the evolution of complexity was not measured quantitatively, it was evident through our qualitative observation that the process was not smoothly incremental. The process, in fact, involves alternating incremental building and reorganisation of the model (see the rising and dropping lines in Figure 6.5). Indeed, the requirements engineer continuously and incrementally develops her conceptual understanding of the problem and represents it in the model—therefore, the complexity grows and the model becomes more and more complex over time. Upon reaching critical points, she reorganises her understanding as a result of a reconceptualising insight. The newly restructured model becomes a basis for further building of conceptual understanding. The triggers for insight have not been identified and reported in the literature to date. Although this identification was not an objective of our study, the pattern of the requirements evolutionary process is revealed clearly through our examination of the record and description of the process.

As we examine closely the critical points at which insight leading to restructure occurs, we can build a preliminary explanation of the pattern in more detail through the lens of the essential and incidental complexities of the model and the dynamics of evolution. Figure 6.6 illustrates in a simple way these two types of complexity and our interpretation of their dynamics. Figure 6.6(a) illustrates the incremental growth of the essential complexity of the requirements model throughout the process. Due to the inevitable complexi...
increasing entropy during the modelling process, the incidental complexity increased exponentially over time (see Figure 6.6(b)). As time progressed, the complex structure of the model made it progressively more difficult to add new components/elements to the model. As a result, the overall complexity of the requirements model was growing gradually (see the gradually rising curve in Figure 6.6(c)). At some stage, the complexity increased rapidly due to the inefficient adding of new information.

The model was then reconceptualised and simplified. The complexity dropped significantly. This is denoted as a rapidly rising curve and a dropping dotted line in Figure 6.6(c).

6.2.2 Building up a catastrophe-cyclic requirement modelling process

Our experience as exemplified above demonstrates that there are two different types of complexity of the specification to be distinguished:
Figure 6.6: A qualitative explanation of the essential and incidental complexity in requirements models

- Essential complexity: this complexity grows as the inherent understanding of the problem develops through the activities of information/knowledge acquisition and analysis. The essential complexity of the model increases throughout the requirements engineering process. This is consistent with the cognitive design process described in the literature as consisting of different activities and frequent movements between the understanding activity and other activities throughout the process (Sutcliffe and Maiden, 1992; Batra and Davis, 1992; Chaiyasut and Shanks, 1994). The growth of the essential complexity towards the “completeness” of the problem complexity confirms Guindon’s statement that the purpose of the understanding activity in RE is “to decrease the incompleteness ...of informal specification” (Guindon, 1989, page 729).

- Incidental complexity: this complexity arises from poor fit between the structure
of the model and the structure of the world which the model aims to represent. This complexity is often regarded as the overall complexity of the model (i.e. not distinguished from the essential complexity). The simplicity (versus the complexity) often measured in a number of constructs used in the requirements model is referred to as a quality factor of the model (for example see Moody and Shanks (1998) and Roseman (1998)). This thesis shows that the incidental complexity does not grow smoothly throughout the modelling process. Indeed, occasional shrinkage of the requirements model in terms of size and/or simplicity clearly indicate occasional reductions of the incidental complexity of the requirements model during the modelling process.

Although both kinds of complexity grow as requirements models are growing, the former is to be encouraged, while the build-up of the later needs to be reduced.

![Complexity vs Time Graph](image)

**Figure 6.7: Catastrophe-cycle requirements modelling process**

From the above analyses, the evolution of the overall complexity of requirements models (i.e. the catastrophe-cycle model) can be described as in Figure 6.7. This process could be explained as follows:
• Initially, our understanding of the problem increases and the complexity is added to the model gradually.

• As the model grows and becomes more complex, poor early modelling choices make it progressively more difficult to add new components to the model. The growth of the overall complexity is slowing down.

• The more complex the model gets, the less efficient it is in accommodating newly discovered information to the model. The overall complexity increases rapidly as a result of either the exponential growth of the incidental complexity or the rapid growth of the problem complexity.

• Finally, the model undergoes a restructure or major modification triggered by radical insight (or sometimes a systematic evaluation of the existing model). The above examples show that at critical points the newly structured model has a new architecture reflecting a new understanding/perception of the problem by the specifier, not simply the structure of previous model being polished or with redundant components removed.

As illustrated above, the two kinds of the complexity evolve in parallel but they are related. Indeed, the evolution (either the increase or the reduction) of the incidental complexity results from the accumulated inherent understanding (including both “systematic” and unplanned discovery of new information) rather than the sole representation/modelling activity preceded by a “complete” understanding. On the other hand, problems caused by poor modelling choices (resulting in incidental complexity) may trigger insight or instigate a formal post hoc re-examination of the model and therefore may spur further growth of the essential complexity.
The refined understanding of the requirements engineering process and two types of complexity in requirements models led me to re-examine closely the use of QOC at the critical points. The examination is presented in the next section.

6.3 Supplementing IBIS with QOC—Increasing essential knowledge

Here I will examine closely what is happening at the points when QOC analyses are structured. Insights gained from this examination also help me to address the question of when to do QOC. One early assumption was that the simplification of the requirements model (the dropping lines in Figure 6.7) reflects incidental complexity being reduced significantly while essential complexity remains static. Let us re-examine the examples in the previous section.

The first example

In the first example, the specifier and her participants were specifying the structure diagram of the argumentation workbench. We were identifying basic classes and specifying requirements for each class in isolation. The complexity of the requirements model grew overtime as more and more components were added into it. An unexpected insight led to exciting progress—the model was reconceptualised: the classes Project, Model and Argument were found to share a common graph structure. As a result, the abstract class Graph was introduced to generate the graph structure for the classes Project, Model and Argument. The model underwent a major restructure and has a new architecture (see Figure 6.3(d)). The discovery of the graph structure
of classes Project, Model and Argument served as a critical turning point (see section 6.2.1 for detailed analysis of the advantages of this discovery).

Although the generation of the insightful idea accelerated the specification process, the participants including the specifier encountered a dilemma when specifying the graph structure—as a binary relationship of vertices $\downarrow\text{Node} \leftrightarrow \downarrow\text{Node}$ or as a ternary relationship of a $\text{Link} \mapsto (\text{Node} \leftrightarrow \text{Node})$ (see for example Issue M below).

Specific Issue M How to specify a graph structure?

Context The model is reconceptualised. The classes Project, Model and Argument are found to share the graph structure. After reading the graph theory in the literature we discuss how to specify a graph in general.

**-Position 1 $\downarrow\text{Node} \leftrightarrow \downarrow\text{Node}$$

AS This directly supports the definition of graph (in general) as a set of vertices and edges between them.

AS In a node-and-link diagram, vertex is a $\downarrow\text{Node}$. 

AO edge is treated as a relation, not a class. Consequently, a $\text{Link}$ of a node-link diagram should be defined with two edges: $(n_1, l)$ and $(l, n_2)$

*Position 2 $\text{Link} \mapsto (\text{Node} \leftrightarrow \text{Node})$

AS This is a more natural expression for the node-and-link diagrams.

AO This expression requires that edge should be considered a class. Therefore, this expression does not support a general graph (where edge is not an entity).

Issues, such as I, J, K, L and M, were retrieved from the IBIS base for the discussion of the graph structure of these classes and the implementation of the insight. Each of these Issues describes only a specific and local view of the problem. Their Positions were considered in the local context of the associated Issues. It was difficult to evaluate the
Q: How to model the graph structure of the classes Project, Model and Argument?

Figure 6.8: QOC analysis created to support the restructuring of the model—the first example

two options—this created the dilemma. A global analysis was needed. A QOC analysis was structured to re-examine the issues and to guide us in implementing the insight.

The QOC analysis (see Figure 6.8) provided a holistic view of the problem and design space around different classes (Project, Model and Argument). By structuring QOC, two modelling options were highlighted and explicitly evaluated against a common set of criteria. The essential complexity of the model increased. In fact, the QOC analysis assisted us in the provision of a global view of the problem and thus in restructuring the model at this crisis point.
The second example

The second example shows that during the modelling process, the participants including the specifier moved between different problem areas when discussing and trying to solve the requirements problem. Although there was a linear stream of issues, these issues concern not only different aspects of the problem, but also reflect frequent backward and forward movements between different problem areas. We moved away from the primary problem of specifying the graph structure to examine the class *Decision* and solved another (but related) problem in this area. Later, we came back to the original problem to refine our understanding of graph structured classes. We not only moved between different problem areas, but also switched between the elicitation and the modelling activities (eliciting new requirements, structuring diagrams, writing Object-Z specifications...). Again, this interpretation of the requirements engineering process is consistent with that offered by Carroll and Swatman (1998) and Carroll and Swatman (1999b) who describe the requirements engineering process as opportunistic. Suddenly, a new understanding emerged: a recursive graph structure of different graphs of different types was identified. In the graph structure of *Project*, each node is a *Model* which is a graph itself. Further, different types of graphs (*Project, Model, Argument*) are associated with different types of nodes and links. The QOC analysis created from the first example (see Figure 6.8) shows two ways for modelling a graph in which links may or may not be modelled as objects. They are illustrated in a simple example in Figure 6.9. The problem turned out to be greater than expected.

According to our specification at this time, the graph structure of classes *Model* and *Argument* links were modelled as objects—a special type of node. However, in the graph structure of the class *Project*, links represent transitions from a version to another were
In the first formulation, links are modelled as objects. The above graph can be modelled as a set of the following pairs of nodes and links:

- \((A, l_1)\)
- \((l_1, B)\)
- \((B, l_2)\)
- \((l_2, D)\)
- \((D, l_3)\)
- \((l_3, C)\)

In the second formulation, links are not formally modelled as objects. The above graph can be modelled as a set of the following pairs of nodes:

- \((A, B)\)
- \((B, D)\)
- \((D, C)\)

Figure 6.9: Different ways of modelling a graph

not modelled as objects. The specifier attempted to structure a complicated hierarchy of different nodes of different graphs (see Figure 6.4(b)). The problem space grew exponentially.

This situation did not persist for long because of a sudden insight: The Model itself could be modelled as a Node. However, the complex nature of the class Model—as a graph of Node and as a Node itself required a thorough examination of the new modelling option. Having learnt from her experience, the specifier recognised that a QOC would be helpful in examining and comprehending the situation. As a result, a QOC analysis was structured for reviewing a number of the IBIS issues associated with this problem (see section 6.2.1 for more details).
This QOC analysis was useful in generating a set of common Criteria and associated Assessments from the IBIS Issues for the evaluation of the two options. Through generating and weighing Criteria and assessing the options against the Criteria, the specifier and her participants reviewed the problem situation from a global perspective. Their requirements/objectives were identified clearly with the first three Criteria. The Criteria C1 and C2 were essential. Initially, the Criterion C3 concerned only a general graph structure of the class Model. Later, as a result of elicitation and discussion, the node-link structure of Model was decided. Criteria (C4 and C5) were also used for assessing the two options, they were particularly important as both modelling options satisfy the two most clearly important Criteria (C1 and C2). C4 is a general criterion while C5 is a specific criterion. Sometimes, general criteria (avoiding redundancy, expressiveness...) are less important than some specific goal-oriented criteria. However, in this
case, a simple hierarchical class structure has a higher priority in order to avoid redundancy and implementation complexity, especially since multiple inheritance is involved in both options. Therefore, after the insight took place, through generating and highlighting Criteria and Assessment, requirements were discussed and negotiated between the specifier and other participants, and the problem space was examined holistically and thoroughly. The essential knowledge increased as a result of structuring the QOC analysis while the incidental complexity was reduced as a result of the restructure of the model.

**A suggested interpretation**

The IBIS issues recorded during the modelling process provide us with the history of this design process. They are discussions between the specifier and two participants. Over time, the recorded IBIS issues form a long stream of arguments (see Figure 6.11). Each argument is structured in a specific context and has its Positions and Arguments. As discussed in Chapter 5, reviewing a large number of different issues recorded at different times for different contexts was difficult. Therefore, occasionally QOC was used as a mechanism to review and reorganise the IBIS issues (see Figure 6.12).

![Figure 6.11: A linear stream of IBIS arguments is formed. Reproduced from Figure 4.3 for readers’ convenience](image)
Our qualitative explanation of the usefulness of QOC in supplementing IBIS at crisis points is illustrated in Figure 6.13\textsuperscript{5}. Although the contribution of QOC does not lie in the reduction of the incidental complexity which would take place due to insight (not the outcome of QOC), the gain lies in the essential understanding of the problem. As exemplified in this discussion, QOC analyses neither cause the sawtooth pattern to happen nor necessarily lead to insights. However, QOC analyses assist us in taking advantage of the expectation of the pattern and the examination and implementation of insights. It leverages the increase of the essential complexity of the model.

Initially, the researcher decided to use QOC periodically when the IBIS base got incon-

\textsuperscript{5}This figure illustrates the reduction of the incidental complexity and the gain of the essential complexity qualitatively. It does not give any measurement in terms of the time dimension.
veniently large, when she encountered a very difficult problem or when she felt the need to do so intuitively. However, as exemplified in the above two examples, the researcher found that she did not create QOC periodically by the indication of size of the IBIS base. She was, in fact, “invited” to use QOC at critical points to take advantage of reconceptualisation insights. The role of insights in reconceptualising the requirements problem and the usefulness of QOC in taking advantage of insights and supplementing the IBIS base led the researcher to revise her process of using both the notations (see Figure 5.3).

![Diagram of process](https://via.placeholder.com/150)

**Figure 6.14: Using both IBIS and QOC within FOOM**

In the original process (see Figure 5.3), IBIS is used during the modelling process to open up issues, lay out arguments and document outcome of requirements discussions. In the revised process (see Figure 6.14), insight is viewed as a trigger and additional input into the conversion of the IBIS base into QOC analyses. In fact, being prompted by a reconceptualising insight, the specifier may structure a QOC analysis to interrogate and reorganise the IBIS base. The QOC analysis is used in isolation for the assessment
of the problem situation and the evaluation of modelling options. The QOC analysis leverages effects of the insight through gaining a holistic understanding.

6.4 Summary

In summary, by recording the RE process using a design rationale notation and analysing the complexity of the requirements models, the oscillations in complexity throughout the process have been traced and reveal the catastrophe-cycle of the requirements modelling process. The process is not smoothly evolutionary, but involves occasional “crisis” points at which the model is reconceptualised, simplified and restructured. A close examination of the evolution of the complexity of the requirements model shows that there are two different types of complexity of the model: the essential and the incidental complexities. The essential complexity represents the inherent understanding of the problem space embedded in the model while the incidental complexity arises from the poor fit between the structure of the model and the structure of the real world problem which the model aims to represent. The evolution of the requirements model involves both the growth of the essential complexity throughout the exploring of the problem space and the growth and shrinkage of the incidental complexity as the model undergoes a large number of changes.

The insight-driven nature of the requirements engineering process has been observed, confirmed and refined. The recorded IBIS base shows an opportunistic movement by the specifier between different problem areas during the process. When a reconceptualising insight occurs, the IBIS base provides essential input for the reconceptualisation, simplification and major restructuring of the model. A post hoc analysis may be restructured using the QOC notation to take advantage of insight. The requirements
problem is reconceptualised, the model undergoes a restructuring and has a new architecture. Its complexity is reduced significantly. Specifically, the incidental complexity is reduced. However, as a result of the QOC analysis, the essential knowledge embedded in the model increased.

The new understanding of the complexity of the requirements model and its dynamics and the usefulness of QOC in gaining the essential knowledge at crisis points indicates new directions for future research and forms a basis for a new approach to process management. Chapter 7 will summarise findings from this thesis, discuss their implications and point out directions for future research.
Chapter 7

Conclusions and Future Research

7.1 Summary

Within this thesis, I presented the rationale, design and process of an action research project which studied the use of design explanation within the requirements engineering process:

**Research rationale** A critical literature review of three themes of Requirements Engineering, Design Explanation and a specific requirements engineering method (FOOM) suggested an intellectual framework in which systematic documentation of the requirements model and underlying arguments could be useful in understanding the evolution of requirements and in monitoring and improving the requirements engineering process. This strongly encouraged me to explore the use of design explanation within requirements engineering though incorporating it within FOOM.

**Research design and process** The process of using design explanation within re-
quirements engineering was observed and interpreted by the researcher through two hermeneutic cycles. Specifically, the researcher first applied the design explanation notation IBIS and later both notations IBIS and QOC within the FOOM process in a requirements specification project. The researcher reflected upon their application and derived a meaningful interpretation and explanation of her observation and experience. The learning from the first research cycle led to a redesign of the process of using design explanation during the second research cycle. Specifically, the strengths and weaknesses of recording the FOOM process using IBIS, identified in the first hermeneutic cycle, led to the supplementing of IBIS arguments using QOC analysis in the second hermeneutic cycle. The learning from the second research cycle confirmed the benefit of complementary use of both the notations within FOOM. These hermeneutic cycles resulted in a new understanding of the requirements engineering process and a new approach (M) to using design explanation within it. The findings also confirmed and extended the framework (F), described in Chapter 3 Page 148.

In this chapter, I summarise the findings of this research, discuss their implications and outline directions for future research.

### 7.2 Summary of findings

This study focused on the two primary research questions, described in Chapter 3 Page 119, which concern the use of design explanation within requirements engineering. These research questions were answered and analysed in detail in Chapters 4, 5 and 6. Interestingly, the study also resulted in a significant new understanding of the dynamic
process of requirements engineering. This analysis was reported in Chapter 6.

These findings are summarised in this section.

7.2.1 Principal research question 1

Is design explanation useful for the description, explanation and hence the “management” of the evolutionary process of a specification?

Both *ad hoc* IBIS and *post hoc* QOC approaches to design explanation were found to be useful for the description, explanation and monitoring of the requirements engineering process. Particularly, IBIS was found to be useful in:

- describing the requirements engineering process
- explaining the requirements evolution
- providing essential input for the reconceptualisation, simplification and major restructuring of the requirements model
- supporting communication between participants

Weaknesses of IBIS, however, were found to include:

- the locality (fine focus) of *ad hoc* arguments and the lack of context
- the difficulty in searching within and managing the large IBIS base

QOC was used to supplement IBIS at “crisis” points with the aim of mitigating these weaknesses. QOC assisted the requirements specifier in:

- reorganising and interrogating the IBIS base
• providing a holistic understanding of the requirements problem and assisting the evaluation of alternative requirements models

• taking advantage of the reconceptualising insights resulting in the gain of essential knowledge

• reducing searching time in the IBIS base.

Both notations were found to be useful in:

• documenting and managing the process knowledge. The IBIS base provides a chronological record of the requirements engineering process while the QOC base provides a mechanism for logical analysis of the problem space structured through reexamining, reorganising and consolidating the large IBIS base.

• supporting better understanding of requirements models written in different forms and their evolution.

7.2.2 Principal research question 2

If design explanation can be shown to be useful, how should it be incorporated within FOOM?

The application of IBIS and QOC within FOOM was shown to be useful and desirable for the requirements specifier. The learning from this research project led to suggestions about how to use both notations within the FOOM process effectively. The suggestions are summarised below in terms of the process, notation, and usage of design explanation.
Process

- Using IBIS and QOC to document and manage the process knowledge. IBIS should be constructed to record requirements decisions as they are made. QOC should be constructed to supplement the IBIS base at crisis points. The analyses in Chapters 4, 5 and 6 show clearly that this complementary use of IBIS and QOC provides an effective mechanism for recording and organising process knowledge so that it becomes more quickly and comprehensively accessible. Figure 6.14 describes this process of using IBIS and QOC.

- Using IBIS and QOC to record and structure the evolution graph of the requirements model and the rationale for the evolution. This thesis leads to a micro-level process model for structuring the graph of requirements evolution and explaining that evolution (Figure 5.12). This process model is described in Chapter 5, Section 5.3.2. The documentation activity is version-centred. Initially, a current version of the requirements model is selected for discussion. A candidate version is created. Discussions about these versions are documented using IBIS. A QOC analysis may be created to review all the IBIS Issues related to the versions. A QOC analysis is not required to be created for each version released, in fact, QOC is rather associated with reconceptualisation insights and is triggered opportunistically (see Page 277).

IBIS arguments and QOC analyses are attached to the FOOM components under discussion. The IBIS and QOC documents having similar effects on a requirements version are grouped in a design decision. Design decisions are classified into the types Documentation and Transition. Documentation
and **Transition** objects are attached to the relevant intermediate specification version. **Documentation** objects hold all design explanation documents which explain and confirm a version and are attached to that version. **Transition** objects hold all design explanation documents which argue for a transition from a model to another and are attached to the relevant transition. This classification of design decisions is discussed in Chapter 5.

In this way, all IBIS arguments and QOC analyses are attached to either a model they support or a transition they argue for. The QOC base provides a summary of (and therefore, a better access to) the IBIS base. Temporal versions are not necessarily stored but can be structured when needed. Therefore, this use of both notations effectively documents both the evolution of the requirements model and its associated rationale.

**Notation**

- As discussed in Chapters 4 and 5, the thesis suggests the additional component **Context** and a flexible use of the IBIS notation. The component **Context** was introduced in the first research cycle. Throughout the two research cycles, it was shown to be useful in describing the situation in which an **Issue** exists while maintaining the essential simplicity of the IBIS notation. Further studies into representing the connections among different **Contexts**, and between this component and other IBIS components, and possibly the requirements model, are desirable. However, this should not be done at the expense of the simplicity. The IBIS notation should be used in a flexible mode so that it minimises effort in creating documents by the requirements engineer and does not intervene in the requirements engineering process.
Usage

- Design explanation should be directly created by the requirements engineer (and by participants at times) to best maintain the connection between the evolution of the requirement model and its underlying rationale, as well as to support their collaborative work. A process of using Design Explanation within FOOM to support collaboration is described in Chapter 5 Page 209.

7.2.3 New understanding of the requirements engineering process and its dynamics

Catastrophe-cycle model of the requirements engineering process

The process was not smoothly evolutionary, but involved occasional “crisis” points at which the model was reconceptualised, simplified and restructured.

My qualitative observation and analysis of the requirements engineering process show that the process involved intertwined activities of the construction of the problem space as well as the generation and evaluation of its workable solutions. During the process, the requirements problem was continuously explored and structured. Components of the requirements model were introduced as new information was being acquired, accumulated and represented. The overall complexity grew overtime.

At the critical point, the problem was reconceptualised, the model was simplified and restructured, the complexity was reduced significantly. The reconceptualisation accelerated the development process and the new architecture of the model elevated the level of abstraction of the model. The newly restructured model became a basis for a further development cycle. A detailed description of the
dynamics of requirement model complexity is illustrated in Figure 6.7.

**Essential and incidental complexity of the requirements model** As discussed in Chapter 6, the evolution of the requirements model involved both the growth of the essential complexity throughout the discovery of new information and the growth and shrinkage of the incidental complexity of the model as the model underwent a large number of changes. The essential complexity reflects the intrinsic understanding of the problem and grows towards the “completeness” of the problem complexity. The incidental complexity represents the complexity of expression/representation rather than substance in the model and grows as discovered information/understanding is modelled and occasionally shrinks as creative insights happen.

**Opportunism and the catastrophe-cycle model** A detailed qualitative analysis of the context of crisis points shows:

- At crisis points, the newly restructured model had a new architecture and reflected a new conceptual understanding of the problem. It was not simply the previous model with a reduced number of components through removing redundant components and polishing the model. The new architecture of the model reflects a new perception of the requirements problem.

- The reconceptualisation of the problem tended to be opportunistic and insight-driven, *i.e.* not through systematic analysis or deliberate effort. The reconceptualisation relies on the requirements engineer’s creativity. This clearly shows that the solving of requirements problems depends on the requirements engineer being creative and flexible in changing his/her perception of the problem and being able to view it from different perspectives.
• Although the overall complexity was reduced, it was the incidental complexity which was reduced, not the essential complexity. At crisis points, a QOC analysis, created to interrogate the problem space, leveraged the gain of the essential complexity.

Therefore, these findings firstly confirm the creativity and opportunistic characteristics of the requirements engineering process, discussed in Chapter 2 Page 21. Secondly, the findings also extend this understanding by revealing the effects of creative and opportunistic insights in problem understanding and solving activity in requirements engineering. Finally, they suggest a way of increasing these effects using a post hoc examination of the problem space. We can say with some confidence that this new understanding of opportunism may apply to other human creative design activities. However, the identification of the trigger for insight has not been an objective of the research and still remains an open question.

7.3 Implications

This section discusses implications of the above findings for FOOM, requirements engineering and design rationale. In addition, this section also discusses an issue in educating professional requirements engineers.

7.3.1 A new approach to using design explanation within FOOM

The usefulness of IBIS and QOC, together with the identification of this process pattern and two types of requirements model complexity, forms the basis for a new approach (M) to monitoring and controlling the process and to supporting creativity in FOOM.
Understanding, monitoring and controlling the FOOM process

The IBIS base provided a description of the RE process. In fact, design meetings and decisions were recorded and coded as they occurred using the IBIS notation. Therefore, the IBIS base explained in chronological order how requirements evolved (see Chapters 4 and 5). Reviewing the state of the practice in requirements modelling, Lubars et al. (1993) discover that designers appear unable to describe how initial informal requirements are transformed into precise specifications. The authors acknowledge that “it is not easy to make specific recommendations about how to improve requirements practices, because there is little evidence about exactly what analysts, developers and marketing people do.” (Lubars et al., 1993, page 2). From experience gained in this research project, I believe that the IBIS base supports managers in understanding and monitoring the growth and shrinkage of the complexity of the evolving specification.

As analysed in Chapters 5 and 6, QOC is useful in supplementing IBIS. At crisis points, IBIS arguments were reviewed and converted into QOC analyses for the evaluation of FOOM models. QOC analyses provided the specifier with the retrospective logical and holistic evaluation of the requirements model at different stages of development.

Therefore, the complementary use of IBIS and QOC enables managers to understand and monitor the evolution of FOOM models.

Supporting reflectivity and creativity

The new understanding of the opportunistic nature of the requirements engineering process and the oscillations in complexity and the use of both IBIS and QOC (above) suggest that design explanation supports reflectivity and creativity in requirements engineering:
As shown in Chapter 6, the IBIS arguments spelled out the reasoning of the participants including the specifier, kept them reflective about their actions, helped to refresh the design memory, and assisted the reaching of shared understanding among all the participants. Our experience, in this thesis, confirms the usefulness of IBIS in reflection-in-action, a concept introduced by Schön (1983) (see Chapter 2, Page 54) and later supported by the requirements engineering and design explanation research community (see for examples Chapter 2 Pages 32 and 72 respectively). Indeed, Fischer et al. (1991, page 282) argue: “design rationale can aid reflection by informing it with the design knowledge, principles, and ideas, and by triggering critical thought in the designer”. In this thesis, reading and examining the IBIS notes also helped the specifier and her participants generate ideas and creative insights. Briefly, IBIS supported reflection-in-action through providing the specifier with an accumulated knowledge of the problem space and assisting her in reflecting on the progress towards achieving the goal.

As analysed in Chapter 5, the IBIS arguments explain how the requirements model developed the way it did while the QOC analyses explain why the model had a certain form at a specific stage. QOC provided the specifier with an understanding of the current status of the requirements model and assisted her in controlling the development process. Therefore, QOC supports what Schön (1987) calls reflection-on-action.

In addition, as analysed in Chapter 6, another significant contribution of QOC lies in taking advantage of reconceptualising insight resulting in the gain of the essential understanding of the problem at crisis points.

Therefore, although design explanation does not directly increase creativity, the IBIS
arguments and QOC analyses and the complexity of the model being monitored enable the specifier and project manager to understand the on-going creative process of requirements engineering and leverage effects of creative insights.

In addition, the inherently creative, opportunistic nature of the requirements engineering process (analysed in Chapter 6) poses problems about how to manage the process without restricting its creativity. The use of design explanation in this thesis supported the FOOM process without undesirably intruding on the process. Indeed, the simplicity and \textit{ad hoc} characteristics of the IBIS notation allowed us to document the process non-intrusively while QOC could be used in solving critical problems and in interrogating and consolidating the IBIS base when needed. This suggests that the requirements engineering process can be monitored and supported without interference in the process or without decreasing the flexibility needed for requirements engineers’ creativity.

Initially, the researcher believed that although the research involved a specific requirements engineering method (FOOM), the research outcome could be ‘directly’ applied to other methods. However, as mentioned early in Chapter 2 Page 116, by revealing a new understanding of the process of requirements engineering, this thesis raises a new question of the appropriateness of current approaches to requirements engineering as they, in general, are based on an incremental evolution of requirements, rather than offering a directly applicable conclusion. The next section will discuss the implications of the research outcome in requirements engineering in general.
7.3.2 Requirements engineering

Do current systems development approaches support the development process as it really is?

As discussed and concluded in Chapter 2 Pages 46 - 50, a deep understanding of the process plays a critical role in process (and project) management. The literature review of various perspectives on the requirements engineering process and systems development life cycles in this chapter shows that most process management approaches are based on the traditional assumptions of an incremental development. Therefore, they are in conflict with the creative and opportunistic model of problem exploration and problem solving activities which appears to be characteristic of requirements engineering (from the discussions in Chapter 6). Particularly, the catastrophe-cycle requirements modelling process strongly suggests that various process management approaches and CASE tools imposing the incremental evolution of requirements models should be reexamined to determine whether they assist or, in fact, they handicap the system developer.

Monitoring, controlling and improving the evolution of requirements and the creative development process

The above findings and the researcher’s reflection upon the intellectual framework \( F \) used for defining the findings of this thesis led to an interesting implication. Design explanation and the dynamics of the complexity being monitored would signal the project/process manager when managerial actions might be needed. Indeed, deviations from the catastrophe-cycle pattern would inform the managers (and the specifiers) of two possible situations:

Lack of shrinkage in the complexity of the requirements model This
cates that there is a lack of flexibility of the requirements engineer in viewing the requirements problem from different aspects and restructuring the model. The modelling activity might be disorganised and the model might be too complex. This might be due to eliciting an excess of information and thus continuously building up complexity without reorganising the information, crystallising ideas and restructuring the model at appropriate stages.

**Excessive frequency of shrinkage in the complexity of the model** There is a lack of persistence and coherence in the modelling activities. The developers might spend too much time on the representation and changing of the model without building up a holistic or deep understanding of the problem, e.g. rushing into expressing and reorganising components of the model without conceptualising and developing a mature understanding of the problem space.

In addition, the recognition of the essential and incidental complexity in the requirements model suggests challenging objectives for researchers who are working to improve the requirements engineering process: to increase the intrinsic complexity in the content and to minimise the incidental complexity due to entropy.

**Encouraging collaboration among various participants**

Findings from the thesis encourage collaboration among various participants including the specifiers during the elicitation, modelling, negotiation and validation of requirements. The role of design explanation in resolving conflicts, evaluating arguments and prioritising criteria of requirements is strongly encouraged. The simplicity of design explanation notations assists the client to input his business knowledge/contribution into the process.
Developing support tools

There are a number of implications for the current effort towards developing automated development and support tools in requirements engineering:

- The new understanding of the process suggests that there is a need to support the development process as it really is. Most of today’s tools impose an incremental evolution of the requirements model, therefore, fail to encourage the generation and evaluation of creative insight and the reconceptualisation of the requirements problem.

- There is a need to monitor the dynamics of the complexity of requirements models and to assist the specifier in being aware of the growth and shrinkage of the complexity and crisis points as the model evolves. The various approaches to managing the requirements volatility do not appear to be based on a deep understanding of the process or a strong theory about the process. Therefore, most of them fail or have not been adopted widely by the professionals. The learning from this research also shows clearly that requirements volatility may not necessarily be ‘bad’, in fact, it might be required as the requirement space is continually structured and restructured. A more appropriate and challenging traceability issue should be: how to support the traceability of the requirements model which might have totally different architectures. A tool which assists the monitoring of complexity of the requirements model would supplement the above complementary use of IBIS and QOC in addressing this issue.

In addition, with regard to the need for supporting both horizontal and vertical traceabilities, discussed in Chapter 2 Page 39, findings of this thesis strengthens the role of design explanation for this purpose. Specifically, design explanation
provides an unified view across different representation forms of requirements, and therefore, should be integrated into traceability tools.

- There is also the need for an integrated environment CASE tool to assist designers in recording rationale and editing requirements models, as prerequisites for managing the process. Tools should be aligned to a specific requirements engineering method to best support links between different representation formats and stages/activities recommended by the method.

### 7.3.3 Design explanation

This research differs from previous research into design explanation. It studies design explanation in the context of a design process—both the application of design explanation and the design process are under investigation. It also shifts the focus of study from a prescribed design explanation approach for general requirements analysis to a selected but not restricted approach and its possible adaptation to suit and improve a specific requirements engineering method, namely FOOM. Specifically, this research does not study QOC in isolation but as a supplement for IBIS. By studying different design explanation approaches in the context of the cognitive process of problem understanding and solving activities rather than focusing on a specific notation, this thesis responds to the issues raised in Chapter 2 (see Pages 73, 93 and 110), particularly the “what, when and how” issue, and offers a new approach to capturing the rationale information and managing the large rationale base accumulated overtime. This thesis leads to the following implications:

**What** IBIS and QOC are qualitatively differed in terms of the problem they address—they can and, indeed, should be used complementarily. IBIS can be used for doc-
umenting the *ad hoc* decision making process as it occurs while QOC can be used for analysing the problem space, reorganising the design ideas and reviewing the design product at different stages. The IBIS base provides an essential input for the construction of QOC. The QOC base, in turn, provides a mechanism for interrogating and reorganising the IBIS base. This addresses the concern of managing and accessing the large IBIS base as well as the concern of maintaining a rough and evolving QOC. Indeed, the new approach ‘frees’ designers from capturing rationalised flying thoughts (rough QOC) when he/she engages in the complex design activity and yet still provides them with both an accessible design history and a rationalised evaluation of the design space when needed.

Therefore, IBIS base provides the *ad hoc* analysis path while the QOC base provides the holistic and logical structure of the problem space at different development stages.

**When** Although design explanation should be used in a flexible manner in order to support the creative nature of design activity, there should be a ‘systematic’ guideline for using design explanation in order to maximise its usefulness. This thesis suggests that IBIS should be created when a design decision is made while QOC should be created after the advent of a significant creative insight which may reconceptualise and reshape the design space. Therefore, the designer is not forced to use QOC continuously or periodically during the design process, but is invited to use QOC opportunistically when needed. Since why and how an insight happens is still an open issue, the creation of QOC cannot be pre-planned or predicted. It depends on the designer’s creativity in constructing and solving the design problem. This suggestion is consistent with current conclusions of
the QOC research community that QOC should not document a complete design space but should be used for solving critical problems. Furthermore, this thesis also deepens this current understanding by highlighting that opportunistic insight triggers and is an essential input for the construction of QOC.

**How** This will be discussed in terms of process, notation, usage and support tools.

**Process** Design explanation should support the design process as it really is. A deep understanding of the design process is essential in using design explanation so that it supports, but does not distract the designer. As the designer is required to be able to switch between different problems areas and between different levels of abstraction at the same time, a complementary use of both *ad hoc* and *post hoc* design explanation approaches is needed at appropriate times. IBIS provides the designer with a mechanism for recording of his/her movements between specific problems areas and design ideas whereas QOC allows the designer to work at a higher abstract level. In addition, it is very important that design explanation should be aligned to a specific design method (see Sections 7.3.1, 7.2 for a process for using IBIS and QOC within FOOM).

**Notation** The thesis confirms the need for representing Criteria weights and priorities in order to support the designer in judging the trade-off analysis between different design alternatives. A flexible use of notation is needed to enable the individual design project to adjust a design explanation notation for their application.

**Usage** Since there are different views on the usage of design explanation in terms of who, this thesis affirms that design explanation can and should be created
and maintained by the designer.

**Tools** Design explanation should be integrated within a specific design method and attached to the design product (and design components) in order to increase the understandability of the design product and explain its evolution. In addition, as design explanation should be used by the designer him/herself, an integrated environment for supporting a specific design method and argumentation notations is highly desirable. This is consistent with previous research into IBIS and QOC.

In addition, since both a complementary use of different approaches to design explanation and a flexible use of their notations is useful and desirable, the need for automated tools which are based on a meta-argumentation model (for example, Shanks et al., 1994) is confirmed.

The thesis not only offers a new approach to using design explanation within requirements engineering, but also suggests issues for future research into design rationale. These include:

- Cost and benefit studies should be conducted for the evaluation of the new approach.

- The applicability of the approach should be investigated in other phases of the systems development life cycle, moreover, in other domains of human problem understanding and solving activities.
7.3.4 How should we train requirements engineers?

Requirements engineering, as revealed in this research, requires both insight and creativity as well as technical knowledge. However, traditional approaches to training requirements engineers, in particular, and the Information Technology professionals, in general, tend to focus on technical knowledge, largely on notations and prescribed processes. The question of how we can (and should) train requirements engineers to work effectively in an environment, where insight and creativity are required, now becomes a central issue in Information Technology education.

7.4 Future research

My research project was intended to generate a qualitative interpretation of the use of design explanation within requirements engineering and to gain a deeper understanding of the requirements engineering process. However, due to the nature of this research project, there are some factors which might limit the generalisability of the research outcome:

- As discussed in Chapter 3, the research method adopted in this thesis is placed towards the interpretivist end of the methodological continuum. Interpretivist research offers advantages in providing a rich picture of a specific context, embracing more complex variables, gathering meaningful explanations of the phenomenon and deriving fruitful insights and concepts strongly grounded in reality. On the other hand, interpretivist approaches are often criticised as being potentially anecdotal and lacking in rigour.

- Although this thesis involved a real requirements engineering project, it was car-
ried out in an artificial setting, i.e. it did not involve a commercial project. The research setting closely reflected complex reality and enabled flexible control which allowed me to pursue research interests as concepts evolved. The elimination of time and cost constraints on the development may compromise the generality of applicability of the results. Clearly, the replication of this research in a real world commercial environment is required.

These issues will be addressed by further work arising from this thesis:

**Consolidating theory** I will relate the catastrophe-cycle model to cognitive studies in order to build sound theoretical bases for the model. Cognitive behaviour and the similarities and differences between novices and expert designers have been discussed in the literature (see for examples, Batra and Davis, 1992; Chaiyasut and Shanks, 1994; Sutcliffe and Maiden, 1992). These authors find that both experts and novices share similarities in cognitive behaviour and dependency relationships between behaviours. Chaiyasut and Shanks (1994), however, find that novices spend most of their time in the problem understanding activity while the experts spent most of their time on the modelling activity. Novices’ models tend to be developed “literally” from the problem description while experts’ models tend to contain “many concepts not explicitly mentioned in the case problem” (Chaiyasut and Shanks, 1994, page 320). Models developed by experts, therefore, are more comprehensive, complete and hold a holistic view of the problem. This stream of research will inform a further investigation of the catastrophe-cycle model in the context of cognitive behaviours.

**Further developing the new understanding** There is a need to study the dynam-
ics of both essential and incidental complexity in relation to networks of different cognitive design activities described in the literature. This study will develop a sound theoretical foundation for our understanding of the requirements modelling process.

**Testing the catastrophe-cycle model and evaluating the new approach to using design explanation** This model was identified from analysing qualitative data. However, quantitative measurements are needed to confirm and strengthen the model. The model will be tested through empirical studies. A quantitative measurement of the complexity to test the qualitative explanation will be conducted. FOOM specifications, being expressed in formal Object-Z specification and object-oriented diagrams, would assist the researcher in measuring the complexity of requirements models. The current plan is:

- To choose an Object-Oriented metrics scheme to measure the overall complexity of the documented static versions of the requirements model. Complexity metrics are hard to define. Complexity is probably best defined in terms of the ease of comprehension. It is thus difficult to define independently of the person doing the understanding. However, as pointed out in Henderson-Sellers (1996, page 53), the most important measure is the ratio of the complexity of the model (here our requirements model) to the complexity of the problem itself. If the model is the optimal description of the real world situation, then the essential complexity mirrors the problem complexity. Initially, the essential complexity is anticipated (as defined above) to only measure part of the problem and thus increase asymptotically towards the value of the problem complexity as time progresses and further
details of the problem are uncovered and modelled. The issue of measurements of problem complexity (especially, with regard to “change over time”) has been rarely discussed in the literature. Thus this future research will advance current understanding in problem complexity in requirements engineering. The growth and shrinkage of the complexity of the model over time will be measured based on the selected metrics and analysed in detail.

• To identify what factors express the effectiveness of restructuring of the model, for example duration and change (in both quantitative units and qualitative assessment). The connection between the measured dynamics of complexity and the experience of the actors within the process will also be taken into account. These factors will be used as a basis for a quantitative examination of the use of design explanation within FOOM.

• To conduct empirical studies to test the catastrophe-cycle model. Longitudinal field studies will be conducted for measuring complexity over time and testing the catastrophe-cycle model both quantitatively and qualitatively. Snapshots of commercial projects will be taken, their complexity will be measured. Data will be subjected to both quantitative and qualitative analyses.

Quantitative analysis The metrics to be used for both the overall and the accidental complexity are expected to be the same—so that the difference will give a measure of “value” for the requirements model. The rate of change with time will be logged and analysed in order to test the hypothesised catastrophe-cycle model.

Qualitative analysis Data will be collected from the requirements mod-
els, produced as a result of the interpretivist, qualitative field work and subjected to complexity analysis. Complexity analysis will be undertaken initially on the basis of the selected metrics scheme, although this scheme may be revised or enhanced in light of the analysis of data collected from the studies.

**Cross-examination of quantitative and qualitative results** The above results will be crossed examined. The catastrophe-cycle model will be evaluated and enhanced.

- To evaluate the suggested approach to using design explanation within requirements engineering in commercial practice. The approach to using IBIS and QOC within requirements engineering will also be evaluated in terms of cost-effective analysis. As discussed in Chapter 6, a quantitative cost benefit analysis of the approach was not appropriate at the present stage but will be conducted at this stage since the approach has been conceptually and qualitatively proved feasible.

The cross-examination between quantitative and qualitative analyses produced from these realistic, longitudinal studies will enable me to consolidate and enhance the new understanding of the requirements engineering process identified from this thesis. The complementary use of IBIS and QOC suggested by this thesis will also be evaluated and consolidated. The results gained from these studies will:

- form bases to develop a new approach to monitoring, controlling and managing the development process using design explanation.

- form bases to develop a new approach to training requirements engineers.
Bibliography


Davies, S. P. (1991). Characterizing the program design activity: neither strictly
top-down nor globally opportunistic, *Behaviour and information technology*


Hall.

methods in Requirements Engineering, *in* A. T. Wood-Harper, N. Jayaratna and
J. R. G. Wood (eds), *Methodologies for Developing and Managing Emerging Tech-
368. Republished from Proceedings of BCS-ISM Conference - 6th Annual Confer-
ence on Methodologies, Salford England, 3-5 September.

Deming, W. (1986). *Out of the Crisis*, Massachusetts Institute of Technology Center
for Advanced Engineering Study, Cambridge Massachusetts USA.

to conduct and report action research. Available on line at

York USA.


Dobson, J. E., Blyth, A. J. C., Chudge, J. and Strens, R. (1994). The ORDIT ap-
proach to organisational requirements, *in* M. Jirotka and J. A. Goguen (eds),


Neuman, W. L. (1994). *Social Research Methods Qualitative and Quantitative Approaches*, 2 edn, Allyn and Bacon, Boston USA.

311


318


an Issue-Based Information Systems, *Proceedings of the Conference on Computer-
Supported Cooperative Work CSCW90*, The Association for Computing Machinery,
Los Angeles USA, pp. 105–118.

Yin, R. K. (1989). Research design issues in using the case study method to study
Management Information Systems, *in* J. Cash and P. Lawrence (eds), *The In-
formation Systems Research Challenge: Qualitative Research Methods*, Vol. 1 of
*Harvard Business School Research Colloquium*, the President and Fellows of Har-
vard College, Boston Massachusetts, chapter 1, pp. 1–6.

Thousand Oaks California USA.
List of publications from this thesis

Refereed Journal Articles


Refereed Conference Papers


Appendix A

A Brief Introduction to Formal Object-Oriented Methodology (FOOM)

This introduction is written by Swatman (1996) and appears in Kilov and Harvey (1996, Chapter 18).
FORMAL OBJECT-ORIENTED METHOD SFOOM

Paul Swatman

Centre for Information Systems Research
Swinburne University of Technology, P.O. Box 218, Hawthorn, Victoria 3122
Australia
Email: paul@swin.edu.au

ABSTRACT

FOOM (Formal Object-Oriented Methodology) is a method which assists in understanding and modelling organisations and thus information systems. The method promotes:

- highly appropriate and precise requirements specifications for information systems
- models of organisational and inter-organisational processes, which may form the basis for organisational and network reengineering

In this paper, I summarise FOOM, which has been under development since 1989 and which was initially described by Swatman & Swatman [SS92a], and argue that the FOOM approach offers clear potential benefits within the Information Systems domain. The paper is non-technical in nature and is designed to offer a summary of existing work and future directions. References to the detailed and technical publications of the FOOM project are provided for the interested reader.
INTRODUCTION

Boehm’s studies of systems development at IBM and TRW [B76], which are still widely accepted, showed that approximately 60% of system faults can be traced to specification errors, ambiguities and omissions. The FOOM project has developed and continues to enhance an information systems requirements engineering method which focuses on the problem of precisely specifying appropriate systems.

This paper describes, in outline, FOOM and provides references to the published literature on the detail of the method for the interested reader. Much of this literature is accessible via the Web.\(^1\)

This paper is structured as follows:

**Background** The motivation for FOOM and the issues which the method addresses

**The FOOM Project** A brief history and a description of the FOOM approach in outline

**The FOOM specification** The format of the output of our requirements engineering method and how this document is validated and used

**Current and future work** Documentation of design rationale and the development of support tools; further evaluation and evolution of the method.

In this paper, I use the first person singular (“I”) to indicate my views as the author of the paper and the first person plural (“we”) to indicate the views of the FOOM project team in general.

BACKGROUND

We were motivated by the dissatisfaction with the performance of the Information systems profession which was pervasive within commerce and industry. The focus of that dissatisfaction is often referred to in the literature as the “Software Crisis” [B87, S87, S92a]. In developing FOOM we have focused on the commercial and industrial rather than the technical and scientific domain. This domain is characterised by clients whose education and focus is primarily non-technical; and by information systems where the complexity of the social and organisational aspects overshadow the technical difficulties.

Computer scientists and software engineers appear to have shown little interest in

\(^1\) The URL for the FOOM home page is: http://mae.ba.swin.edu.au/~paul/foom.html
studying the peculiarities of systems development within this domain. Most research to date has been undertaken by the Information systems community and the foci of interest and thus the approaches taken by these communities of researchers are quite distinct [SS92a]. We have synthesised a method for understanding and modelling problem situations and for specifying and validating the requirements of intervention in those situations, based on ideas developed within both communities. Our method, FOOM, is designed to assist in the process of developing systems which are:

- conformant to requirements
- buildable
- maintainable

**Conformance**

Conformance to requirements is a central goal of any programme of high-quality information systems development. We consider this concept in two parts:

- conformance of the specification to the requirements of the clients; and
- conformance of the resulting system to the specification

It is important that the specification appropriately and accurately represent the problem context, as well as the requirements of and constraints on any proposed intervention. The positivist, objectivist philosophy, which (largely unacknowledged) underlies much of the software engineering community’s research into systems development methods, has been widely criticised within the information systems literature. The critics argue that in socio-organisational contexts there exists no single, objective statement of requirements just waiting to be uncovered, but rather that such contexts may only be viewed subjectively through the cognitive filter of the various associated actors (see, for example, [C81, C95]).

A number of approaches to understanding problem situations which take into account these ideas have been embodied in systems development methodologies, including: Soft Systems Methodology [C81, C89, C95, CS90], ETHICS [MW79] and Multiview [AWH86, WHAA85].

In developing FOOM, we have been strongly influenced by these ideas which enhance the process of eliciting information and understanding of problem contexts and thus contribute to improving the conformance of the specification. Problems do, however, remain. No existing requirements engineering methods which adopt these ideas deliver specifications which are formal (in the sense that they are mathematically precise). The questions, then, are:

- How can we validate a requirements specification which has no precise meaning?
- How can we reliably design and implement a system which fulfills a specification which has no precise meaning?

These questions are, of course, unanswerable. An imprecise specification cannot form
an adequate basis for communication or agreement between the modellers and the actors in the problem domain, nor may it provide a basis for communication with those who will design and implement a specific intervention ("solution"). In principle, the problem of precise representation has long been solved. All engineers understand the necessity and benefit of building mathematical models both for analysis and communication and, within the software engineering research community, formal specification has been the focus of considerable interest for some time. In recent years, we have seen increasing use of formal methods in the development of safety-critical software systems particularly. If we are to build high-quality information systems, it is clear that the specification must be sufficiently precise to provide a baseline for the evaluation of conformance. Consequently, requirements in FOOM are represented in a mathematically formal specification language.

Formal languages allow us to be precise in our communication, but do not necessarily allow us to be effective. For example, the monolithic COBOL and FORTRAN programs of the 1960s were precise specifications, but they quickly became difficult to maintain because, although they were effective mechanisms for communicating with compilers, they were ineffective for communicating with people. We must be able to communicate the specification to the actors in the problem context and the designers/implementors of the consequent intervention. We seek:

- to reduce cognitive dissonance between the model which underlies our specification and the models in the minds of the domain actors and the design/implementation team
- to assist the design implementation team to build their own mental model of the required system.

We have addressed these goals by:

- adopting object-orientation as the underlying abstract architecture for FOOM specifications. The OO community believes both that we naturally understand the world in terms of systems of interacting objects and that OO systems development is seamless in the sense that requirements, design and implementation models for a given system are, typically, architecturally similar (see, for example, [HSE94]).
- incorporating a object-oriented diagrammatic notation (drawn from MOSES [HSE94] then extended and modified by Fowler et al. [FSW95] and Wafula & Swatman [WS95a, WS95b]) as descriptive documentation.

The adoption of an object-oriented abstract architecture also allows us to emphasise reuse. Reuse of well-defined high-quality components reduces some of the opportunities for error in the design/implementation process, provided that appropriate components may be identified. Naturally, reuse of a domain artefact (together with its design and implementation) offers more leverage than reuse of a design artefact which, itself, offers more leverage than simple reuse of an implementation artefact.

One significant problem remains. The conventional wisdom has been that mathematically formal techniques would be unacceptable and, in any event, ineffective...
within the commercial IS context. In our work we have challenged this belief, successfully using formal specifications supplemented by explanatory diagrams, text and discussion in collaborative requirements engineering projects within conventional commercial contexts.

Buildable Systems

We see three important trends in software development occurring at an organisational level:

- the move to third-party systems development [B86,D87, HM84, RH88, SM86, SS90, SSE90]
- system assembly from variously sourced components [A90, G90, M89, R91, S90, VK90]. In these, the development (the design and/or the implementation) of individual system components is sub-contracted (sometimes under the control of a specialist systems integrator, sometimes under direct in-house control) to a variety of third-party organisations. In the general case, some components may be developed in-house. When integrated, these components form the final system
- the increasing importance of inter-organisational information systems [F87, RE88, SS92c].

These trends clearly suggest that, increasingly, information systems will be developed in a complex, multi-organisational context where there may be multiple client organisations and where the “solution” designer(s) and implementor(s) may be organisationally independent of any client. We must, therefore, adopt a requirements specification architecture within which the design and implementation of precisely described components may be subcontracted in a manner analogous to that adopted in more conventional engineering domains. We suggest that formally precise specifications based on an Object-Oriented architecture address this problem.

Maintainable Systems

Systems do, of course, change during their lifetime. Estimates of the ratio of maintenance/enhancement cost to development cost of systems range up to 2:1 and beyond. OO is argued by its proponents to be an intrinsically more maintainable architecture than the common alternatives (see, for example, [HSE94]) and we accept these arguments which appear to be theoretically sound, though I am not aware of any conclusive empirical evidence in support.

We have more recently begun to grapple with a further aspect of maintainability that of understanding existing systems. Our work is incomplete but I will, nonetheless, outline the concepts here. In our earliest work, we considered the problem of communicating our specifications to the domain actors on the one hand and to the designers and implementors on the other. We are only now considering communication.
with maintainers and enhancers of the system.

We see the difficulties here as being similar in many ways to those which face:

- the developers of reuse libraries
- the developers of corporate data models.

[C96] and [S96] have shown that one of the difficulties with ongoing use of corporate data models is that potential reusers are unable easily to understand the models. A potential solution to the problem in this case is the annotation of the data model with design rationale that is, to include documentation which illustrates the reasons which underlie the specific architecture of the data model.

**HISTORY OF THE FOOM PROJECT**

[SS92a] describe, in outline, a framework for an information systems development methodology, widely applicable within the conventional information systems domain, which draws upon a number of established areas of research:

- **Socio-organisational contextual analysis** following the work of Checkland ([C81, C89, C95] and Checkland & Scholes [CS90] which, in the general case, denies the existence of a single, objective requirements specification waiting to be discovered by the systems analyst.

- **The object oriented approach** in which situations are modelled as systems of interacting, encapsulated objects, each object belonging to some class.

- **Mathematically formal specification languages** in particular, the object oriented specification language Object-Z [DKRS91] by means of which the abstract characteristics of classes may be described precisely and unambiguously.

Figure 1 depicts a process model, developed by Fowler [F96], based upon the initial FOOM framework.

All three areas of research make a contribution to the systems analysis and requirements modelling process. The socio-organisational approach is *relevance-centred*. It is concerned with increasing the likelihood that the impact of intervention within the organisational context would be understood – and, thus, that any intervention would be appropriate.

Although the major contributions promoted by adherents to the object oriented approach are concerned with engineering aspects of the systems development process (such as modularity, information hiding, robustness, reuse, traceability), the approach also offers a contribution to communication between specifier and client, to the extent that object orientation is a natural modelling paradigm (see, for example, [HSE94]).
Formal Specifications were, initially, considered to be a conformance-centred approach to software development. They were presented as a basis for a formal “concretisation” process leading to provably correct code. More recently, formal specifications have come to be acknowledged for their contribution to problem understanding [H90, S92a, SS92a, W90], their precision offering potential for enhanced communication and evaluation of understanding.

Our preliminary research [FSS93, S93, SFG92, SS92a] suggested that a socio-organisational approach could be used beneficially in concert with object oriented formal specification techniques within the information systems domain.

The FOOM framework was developed by synthesis and logical argument, drawing on research in a number of largely independent areas across the breadth of the information systems and software engineering domains [SS92a, SS92b]. Preliminary evaluation of feasibility and potential benefit were undertaken by means of simulated [SSD91] case studies and small commercial [SFG92] system specifications; and by means of educational case studies and pseudo-laboratory experiments [S93].

Understanding and Modelling Problem Domains

---

This process is normally referred to as refinement within the specialist literature (see, for example, [M94]. In this paper we use refinement in its more common sense of clarification or increasing relevance.
In common information systems development practice, a problem context is first informally and then systemically specified. The informal specification is ambiguous, imprecise and contradictory. Various systematic specification techniques\(^3\) have been designed to reduce this ambiguity, imprecision and contradiction. Unfortunately, while systematic techniques assist in this process, only specifications written in formal languages which have a well-defined syntax and semantics are capable of precise analysis.

Once the solution design and implementation have occurred, the formalised information system which results has a well-defined behaviour, free of ambiguity. Programming languages are, in a sense, mathematical specification languages and a program listing (in, for example, C or COBOL) is, therefore, a precise and complete specification of the behaviour of the corresponding executable – all programs do something, the question is whether or not this something happens to match the intention of the specifier.

In essence then, the systems development process takes an ill-structured and possibly contradictory context as input and delivers, \textit{inter alia} a precise, well-defined (but, too often, neither the desired or expected) computer system as output. The critical issue, therefore, is to determine at what point we should resolve the imprecisions, ambiguities and contradictions of the problem context. The FOOM approach, which is illustrated in Figure 2, is to resolve these issues during the information analysis process. The argument, set out in detail in [SS92a], is based upon the following assertions:

\begin{itemize}
  \item we must formalise our approach sometime (a program is mathematically precise – if somewhat difficult for human beings to analyse)
  \item it is significantly more likely that the participants in the information analysis process (which would include actors within the context) would bring problem-context-related worldviews to the identification and resolution of ambiguities and contradictions than would software designers and computer
\end{itemize}

\(^3\) For example, Dataflow Diagrams [D79]
programmers whose worldview is likely to be essentially technical

a formal specification of requirements may be communicated to the designers
and programmers precisely and unambiguously.

Following an extensive action research undertaken in collaboration with the Western Australian Department of State Services (DoSS), some aspects of which have been reported by Swatman et al. [SFG92] and Fowler et al. [FSS93], both the FOOM process and the model (requirements specification) which forms the deliverable from that process have been refined. Fowler & Swatman [FS96] describe the process in detail. This action research project has provided strong indicative support for an approach in which OO Analysis diagrams drawn, in modified form, from MOSES [HSE94] are used with text to document the definitive Object-Z specification.

We have therefore developed a semantic correspondence between Object-Z and modified MOSES OO analysis notation. The complete mapping may be divided into three main parts:

A **structural model** which involves mapping the classes and the various structural relationships between them (in particular, inheritance, association and aggregation) described within an Object-Z specification on to MOSES diagrams [WS95b]

An **object communication model** which associates, with the Object-Z specification, a pictorial representation of message passing between the objects and, therefore, control [WS95a]. The pictorial representation corresponds to the notions of interaction graphs [CABDGHJ94], event models [HSE94], interaction diagrams and object diagrams [B94]

A **dynamic model** in which Event Chain notation [FSW95] is used to illustrate and lead the reviewer to a full understanding of the Object-Z fragments extracted from the formal specification which combine to define unambiguously the dynamics of the system.

4 **THE FOOM SPECIFICATION**

The DoSS action research project has also provided strong indicative support [FSW94, FS94] for an approach to specification presentation in which, firstly, requirements specifications are divided into several sections, each having a different focus, to cater for the needs of different audiences and, secondly, OO analysis diagrams drawn, in modified form, from the systematic OO methodologies are used with text in an explanatory role.

The FOOM approach suggests that requirements specifications be presented in the following form:
An executive summary forms the first part of the specification. Designed for presentation to management, this section contains a textual explanation of the system at a very high level, supplemented by informal diagrams.

A behavioural perspective is presented in the middle section, in which Event Chain notation [FSW95] is used to illustrate and lead the reviewer to a full understanding of the fragments extracted from the formal specification which combine to define unambiguously the dynamics of the system. In essence, the reviewer is assisted to consider the specification of the system from a behavioural perspective, firstly at an intuitive level by means of the Event Chain diagrams, then formally, precisely and unambiguously by means of appropriately combined fragments of the complete Object-Z specification which is contained in the final section. The behaviourally oriented section is designed for those user(s) responsible for accepting the specification.

The formal specification in its entirety is presented last and is intended for those responsible for taking the requirements specification and developing the resulting system. The specification is presented in the traditional fashion, with text and OO Analysis diagrams used to explain the mathematics. The development of a well defined mapping from Object-Z into MOSES [HSE94] notation extended by Wafula & Swatman [WS95a, WS95b], has been the subject of recent work by Wafula [W95].

SUMMARY AND FUTURE WORK

The FOOM approach, outlined by Swatman [S92c] has been refined by Fowler [F96] through application in a multi-organisational development of a major inter-organisational information system. Version 1 of FOOM is now documented in [FS96]. Our research has demonstrated that the FOOM approach can be used, apparently beneficially, in typical IS environments but further evaluative work is clearly required. To date, FOOM has been applied only in small or artificial systems developments as the subject of case studies on the one hand; and in a realistic project, but within an action research framework, on the other hand, in which the method evolved as a result of the identification of difficulties and opportunities.

The next phase of the evolutionary process must, we believe, be a series of realistic field trials. We reject the usefulness, certainly at this stage of the development of FOOM, of controlled experiments since our primary concern is to confirm the existence of potential practical benefits from the use of the methodology. There are, however, technical difficulties to be overcome before such realistic case studies may be undertaken:

Object-Z, the formal notation which underlies FOOM, contains many mathematical symbols and conventional word processing software offers only limited support for the creation of Object-Z specifications. In our research to
date, we have used a set of macros which were developed at the University of Queensland to extend the LaTeX typesetting system. This technique, while valuable so far, would, we believe, be totally unacceptable in the professional information systems domain. Some research has already been undertaken into the design of an appropriate user interface to Object-Z [S92b, W94].

The presentation of the requirements specification resulting from the application of FOOM, discussed above, requires that the formal specification not only be presented in full (in section three of the requirements specification) but also that fragments be extracted from it and also presented in section two ... perhaps more than once. It is of course essential that the specification remain consistent! Further, the OO diagrams which support the reviewers' understanding of the formal specification must also remain consistent with the formal specification. Initial work in this area [T94, W95] suggests that it is possible to mechanise the maintenance of consistency to a significant degree and, in any event, to mechanise consistency checking.

Tan [T94] has created the outline design of a FOOM workbench, both to support the creation of Object-Z specifications and associated explanatory diagrams and to support consistency maintenance. Work has recently commenced on the development of a prototype based on this outline design and we hope to begin field trials in late 1996.

Further work on the documentation of design rationale within FOOM specifications (and the benefits thereof); and on quality within the FOOM analysis process itself is currently being undertaken within the Centre for Information Systems Research at Swinburne University by Nguyen and Carroll respectively.

REFERENCES


Formal Object-Oriented Method—FOOM


Formal Object-Oriented Method—FOOM

conformance ............................................................. 299, 300, 303
FOOM .............................................................. 297-300, 302-307
formal specification ........................................................... 300-307
Object -Z ................................................................... 305-307
requirements engineering process ................................................. 302, 304
socio-organisational analysis ....................................................... 299, 302, 303
specificatio validation ............................................................. 305