C 308

MONASH UNIVERSITY
THESIS ACCEPTED IN SATISFACTION OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
ON..... 12 April 2002 .....
.....[redacted].....
Sec. Research Graduate School Committee

ADDENDUM

Page 3, first dot point, should read:

- to determine the empirical validity of the conceptual process model, which is initially grounded in the existing requirements engineering literature, by conducting in-depth case studies of object-oriented requirements engineering which progressively refine and revise the conceptual process model so that the final version of the model is also grounded in current practice.

Page 9, last sentence, should read:

"The process was concluded when a representative model of the research domain being investigated was produced."

Page 10, add after fourth dot point:

"These contributions have emerged from addressing the main aim of the thesis as outlined in section 1.3 above. This aim is overarching and encompasses the specific research objectives, and achieving the objectives led directly to the contributions."

Page 80, add before the last paragraph:

"The logic for multiple case studies is based on the principle of theoretical replication, i.e. replication of results through a series of rigorous case studies (Yin, 1994). The emphasis in this project is on replication in the form of cumulative iteration. A multiple-case study design was used in order to allow cross-case comparison and to strengthen the research findings in the way that multiple experiments strengthen experimental research findings (Benbasat et al. 1987; Yin 1994, p. 31, 45)."

Page 89, last sentence, first paragraph should read:

"The final version of the conceptual process model makes a significant contribution to the theory of object-oriented requirements engineering."

Page 90, Section 4.2, first paragraph: the reference to Simons(2000) should read Simons(1998)

Page 106, add before first paragraph:

"The initial conceptual process model presented below is based on the same three processes and their interactions as Loucopoulos and Karakostas' framework but within the specific context of object-oriented requirements development which includes object-oriented modelling and explicit transformations, flows and interactions."

ERRATA

page 59, last sentence: "The first two models are based on …" should read "The first model is based on …"
page 241, third dot point, paragraph 2: "reuirements" should read "requirements"

An Investigation of the use of Object-Oriented Models in Requirements Engineering Practice

Linda Louise Dawson

B.Sc. (Syd), Grad. Dip. Eng. Dev. (UNSW), M.Sc. (UTas)

This thesis is submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

School of Information Management and Systems

Faculty of Information Technology

Monash University

May, 2001

Abstract

In many organisations there has been increasing use of object-oriented methods for the development of information systems. There is little research into the use of object-oriented methods by practising professionals in producing requirements specifications for commercial or industrial sized projects.

Requirements specification is the starting point for system development. Before a system can be designed and built, it must be specified in a way that facilitates its design and construction. The requirements specification, the product of the requirements engineering stage of systems development, specifies what needs to be designed rather than how it is to be designed. It has been generally recognised that many of the errors that lead to costly maintenance and/or failure of information systems can be traced to omissions, inconsistencies and ambiguities in the initial requirements specification.

Many object-oriented methods and methodologies have been published, based on an underlying paradigm of objects, classes and inheritance, but little has been published about how object-oriented methods and models are used by practising professionals, particularly in the requirements engineering phase of system development. Object-oriented analysis and requirements specification is a relatively new addition to most object-oriented development methodologies. Modelling of requirements is an important part of the requirements specification process. Object-oriented models are said to provide a natural way of modelling real world systems which can incorporate both static or descriptive characteristics and dynamic or behavioural characteristics within one model.

This research project presents a body of work which builds a conceptual process model that represents a theory of the object-oriented requirements engineering process. The model is based on the requirements engineering literature and is refined using data collected during a multiple-case study of object-oriented requirements engineering in practice. The study examined the concepts of the conceptual process model in order to investigate their empirical

validity. Analysis of the data from the six cases in the study has been used to refine the conceptual process model and to identify important implications for object-oriented requirements engineering in practice.

Acknowledgments

I wish to express my gratitude to Peta Darke, for her insightful comments, support and encouragement and for taking on the supervision of this thesis in its final phase.

I would also like to thank

- My colleague Simon Milton for his friendship and discussions about this project and research in general
- My colleagues at Monash University and Deakin University for their support
- Paul Swatman for his comments, especially on early drafts and presentations of this work
- My colleagues in the wider information systems community for their encouragement and comments

Lastly, I would like to thank my husband, Ralph, for his patience and support during the production of this thesis.

May 2001

Declaration

I certify that this thesis does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text.

[redacted]

2nd May 2001.

List of Tables

List of Figures

Table of Contents

Chapter 1

Introduction

1.1 Overview

The increasing use of object-oriented methods for information system development has led to a need for the development of object-oriented approaches to requirements engineering. Although there are many object-oriented methodologies available for systems development which concentrate on design modelling and implementation (Graham, 1994, Booch, 1991, Rumbaugh et al., 1991, Meyer, 1988), object-oriented analysis and requirements specification is a relatively new addition to most object-oriented development methodologies. Many object-oriented methods have been published, based on an underlying paradigm of objects, classes and inheritance (Wegner, 1987), but little has been published about how object-oriented methods and models are used for requirements specification by practising professionals.

Requirements specification is usually the starting point for system development. Before a system can be designed and built, it must be specified in a way that facilitates its design and construction. The requirements

specification, the product of the requirements stage of systems development, specifies " *...what needs to be designed rather than how it is to be designed*" (Macaulay, 1996).

One of the reasons for developing a requirements specification (Loucopoulos and Karakostas, 1995) is that it may form "*... part of the contractual arrangements ...when an organisation wishes to procure a system from some vendor rather than develop it 'in house'*" and many consulting organisations will not give a fixed quote (or "go into contract") for system development unless the requirements specification is as complete and unambiguous as possible. Although requirements may change during the development process, if changes are small then some fine tuning can usually be undertaken without compromising the specification on which the design and construction is based. However, if changes are substantial then the cycle may need to start again (Macaulay, 1996). Incomplete, inadequate, incorrect or unclear requirements have been identified as a major factor in information systems failures (Macaulay, 1996, Boehm, 1981, Lyytinen and Hirschheim, 1987, Jackson, 1997, OASIG, 1996).

The overall objective of this thesis is to explore and describe the use of object-oriented models and methods in requirements engineering practice.

The specific research objectives of this thesis are:

- to investigate and better understand the use of object-oriented methods and models in professional requirements engineering practice
- to propose a conceptual process model for object-oriented requirements engineering in practice based on the relevant literature

- to determine the empirical validity of the conceptual process model by conducting case studies of object-oriented requirements engineering in practice
- to document and describe the findings that emerge from the case studies in terms of the implications for practice and for training.

This chapter presents the background in the domains relevant to this research project and discusses the motivation for this research project. The research approach adopted is briefly described, and the structure of the thesis is outlined. A summary of the main contributions of the research is presented at the end of the chapter. In this thesis the terms "requirements engineer" and "systems analyst" will be used synonymously. The term "methodology" will be used in the manner defined by Avison and Fitzgerald (1995) to mean a collection of techniques and tools used in a particular definition of system development activity rather than in the strict definition of "the science of method" (Concise Oxford Dictionary, 1976).

1.2 Background and Motivation

Requirements engineering may be considered to be a subsidiary discipline of software engineering (Sommerville and Sawyer, 1997). Software engineering developed as a discipline to address the need for developing large complex software systems in an organised and structured manner (Sommerville, 1996). The goal of requirements engineering in this context is to produce a set of system requirements which is as correct, complete and consistent as possible (Sommerville and Sawyer, 1997). Requirements engineering (Loucopoulos and Karakostas, 1995, Macaulay, 1996) deals with the early phase of software development where requirements are determined and expressed as a requirements specification. The requirements engineering process also addresses the problems associated with errors in the specification of

requirements which result in the costly maintenance or failure of software systems (Boehm, 1976, Jackson, 1997, OASIG, 1996), and it is now an accepted term within the information systems community for defining the process and outcome of systems analysis in general (Sommerville and Sawyer, 1997).

Various definitions of and approaches to the requirements engineering process are suggested in the literature. Kotonya and Sommerville (1998) suggest that each organisation must develop its own process which is appropriate for the type of systems it develops, its organisational culture, and the level of experience and ability of the people involved in requirements engineering.

Macaulay (1996) similarly suggests nine different approaches which vary depending on the interest of the groups from which they originate. Pohl (1993) proposes three dimensions of requirements engineering: the specification dimension, the representation dimension and the agreement dimension. Loucopoulos and Karakostas (1995) suggest a specifically process-oriented framework in which the requirements engineering process is broken down into the three sub-processes of requirements engineering: elicitation, specification and validation.

Object-oriented approaches to requirements engineering have generally been embedded in object-oriented system development methodologies rather than existing as stand alone methodologies in their own right. Object-oriented models and methodologies (Booch, 1994, Coad and Yourdon, 1991, Henderson-Sellers and Edwards, 1994, Meyer, 1988) are claimed to provide a more natural way of specifying, designing and implementing information systems based on features which include:

- the ease of understanding object-oriented models due to a consistent underlying representation throughout the development process

- the ability to model the behaviour of objects and encapsulate the static or descriptive characteristics together with the dynamic or behavioural characteristics in a single paradigm
- the ease of modification and extensibility of object-oriented models
- the ease of reuse of object components from previously designed systems
- the incorporation of high-level data abstraction facilities including inheritance and polymorphism.

In this research project, empirical evidence for or against these perceptions is sought which is grounded in the professional use of object-oriented methods on a commercial scale and in a commercial setting.

A major focus of this project is the examination of the use of models in the requirements engineering process. In an object-oriented specification several models are usually produced. These models can be loosely categorised as either static models or dynamic models. *Static models* describe objects, their characteristics and the relationships between them. Some common models are class and object diagrams (Booch, 1994), component notation and templates (Coad and Yourdon, 1991), object models (Rumbaugh et al., 1991), class cards, hierarchies and collaborations (Wirfs-Brock et al., 1990), object/class models (Henderson-Sellers, 1997), and object and layer models (Graham, 1994).

Dynamic models define states of objects, state transitions, message passing and event handling. Some common dynamic models are state transition and event diagrams (Booch, 1994), state diagrams (Coad and Yourdon, 1991, Rumbaugh et al., 1991), objectcharts (Henderson-Sellers, 1997), interaction diagrams (Jacobson et al., 1992), rules (Graham, 1994), and object communication models (Shlaer and Mellor, 1991). Sequencing is often modelled using use cases (Jacobson et al., 1992), task scripts (Graham, 1994) or scenarios (Henderson-Sellers, 1997, Rumbaugh et al., 1991) defining typical user interaction with the system.

Although there has been little research about how practising professionals use object-oriented methods and models for requirements determination, there is some evidence that untrained users have difficulty in understanding the standard data models and object/class models that many professional analysts use during requirements and system modelling (Flynn and Warhurst, 1994, Vessey and Conger, 1994, Weidenhaupt et al., 1998). Various approaches to representing requirements in a manner that is understandable to untrained users have been suggested, often based on use cases and scenarios. Jacobson designed use case models (Jacobson et al., 1992, Jacobson, 1995) as an end user's view of an application which assists in understanding an application's requirements. A use case is a description of a sequence of actions constituting a complete transaction in an application. The use case is a key modelling construct in many object-oriented system development approaches (Jacobson, 1995, Jacobson et al., 1999, Jacobson and Christerson, 1995). A use case can take either a textual or diagrammatic form. Closely related to the use case is the concept of using scenarios (Kotonya and Sommerville, 1998, Weidenhaupt et al., 1998) and task scripts (Graham, 1994) to describe transactions and requirements independently of design and implementation issues. Weidenhaupt et al (1998) found different forms of scenarios such as narrative text, structured text, diagrammatic notations, images, animations and simulations were used in current practice.

There has been little research to date related to the use of object-oriented models and methods in practice. Research in the use of object-oriented specification or analysis models and methods has been mainly limited to studying students in laboratory-type situations. These have usually been studies of small groups where the participants use object-oriented methods for small one-off exercises and problems (Guindon, 1990, Chaiyasut and Shanks, 1994, Boehm-Davis and Ross, 1992, Lee and Pennington, 1994, Morris et al., 1996, Sutcliffe and Maiden, 1992, Vessey and Conger, 1994). However, the behaviour of students, even graduate students, cannot be considered to be indicative of the

behaviour of experienced professionals. Also, the problems used in these experiments tend to be small and, therefore, unlike commercial or industrial projects. Further, some case study evidence in the literature (Carroll and Swatman, 1997) and some laboratory-based empirical evidence (Khushalani et al., 1994) indicates that professional specification (in other fields, such as civil engineering and architecture (Schön, 1983), as well as in requirements engineering) tends to be highly individualistic and often opportunistic. This evidence suggests that specification activities are often based on professional experience acquired over a number of years and a number of projects.

Some studies (Chaiyasut and Shanks, 1994, Guindon, 1990, Sutcliffe and Maiden, 1992, Vitalari and Dickson, 1983) have looked at how analysts apply general problem solving and reasoning skills to the process of requirements engineering, while other studies (Boehm-Davis and Ross, 1992, Lee and Pennington, 1994, Morris et al., 1996, Vessey and Conger, 1994) have addressed specific aspects of using and learning object-oriented methods compared to other methods.

In order to understand *what* successful professional developers do, *how* they do it, *why* they do what they do and *when* they do what they do, further investigation is needed. This thesis presents a body of work which builds a conceptual process model and associated theory of the object-oriented requirements engineering process based on the literature. The model is then refined using data collected from practising professionals. The research project specifically builds on existing research which proposed a conceptual process model (Loucopoulos and Karakostas, 1995) and a requirements engineering framework (Pohl, 1993).

1.3 Outline of the Research Design

The main aim of this research project is to investigate and better understand the use of object-oriented methods and models in professional requirements engineering practice. This aim is encompassed in the following broad research question:

"How are object-oriented modelling methods used by practising professionals in the process of requirements engineering?"

This broad research question can usefully be broken down into three subquestions, based on the three processes of requirements engineering as identified by Loucopoulos and Karakostas (1995) and outlined below:

*Is **elicitation** influenced by the use of object-oriented modelling methods?*
*When, how and for whom is **object-oriented modelling** undertaken?*
*How is **validation** performed on object-oriented models?*

A detailed discussion of the research design and the selection of a research approach used to address these research questions is presented in Chapters 3 and 4. Figure 1.1 illustrates the structure of the research project and shows the relationships between its components.

The first phase of the project is a conceptual study (Galliers, 1992, Shanks et al., 1993). The outcome of the conceptual study is a conceptual process model of the object-oriented requirements engineering process. The term "conceptual process model" used in this thesis is a descriptive term for the model developed from a conceptual study investigating the processes of object-oriented requirements engineering. It draws on and represents similar concepts as Macaulay's general process model (Macaulay, 1996) and Loucopoulos and Karakostas' framework for requirements engineering processes (Loucopoulos

and Karakostas, 1995). This conceptual process model is developed specifically to reflect the general requirements engineering process and the object-oriented requirements engineering process in particular in terms of what the literature indicates is or should be happening in practice. The conceptual process model describes the process-oriented elements of the object-oriented requirements engineering process and is complemented as a foundation of the research design by the research questions proposed above which explore the opinions, beliefs and behaviours of professional systems analysts. The conceptual process model and the research questions are used in the research process to guide the design of interview scripts and data analysis techniques in the case study fieldwork.

The second phase of this project is fieldwork based on a research approach which uses multiple sequential-case studies (Cavaye, 1996, Eisenhardt, 1989, Yin, 1994). This research component uses the conceptual process model and the research questions, which are all grounded in the literature, to develop a set of six sequential-case studies which are used to empirically validate the conceptual process model and answer the proposed research questions. The sequential nature of the case studies allowed for the active exploration of concepts and characteristics of the requirements engineering process which emerged during the course of a case study. Emerging characteristics were then explored further in subsequent case studies, thus producing an accumulation of data over several case studies and refinements to the conceptual process model. Reflection on and re-examination of data collected between cases led to learning and then to revised case documents and interview scripts. The process was concluded when a representative model of the research domain being investigated was believed to have been produced.

1.5 Contributions

This thesis describes the first significant study of the use of object-oriented models and methods in requirements engineering practice. The investigation is based on the refinement of an initial conceptual process model grounded in the literature using the findings of six in-depth sequential-case studies of practising professional requirements engineers. The thesis makes contributions to the theory of object-oriented requirements engineering and requirements engineering in general.

The specific research contributions of this research project are:

- a conceptual process model of the use of object-oriented models and methods which describes and organises key concepts and their relationships in the object-oriented requirements engineering process. This model is grounded in the literature and refined by subsequent case studies of current practice.
- a theoretical model based on the findings which assists in explaining current requirements engineering practice and the implications for practice and training in requirements engineering
- the formulation and demonstration of a research approach specifically tailored to this research project but which provides a useful research framework for similar theory building research, particularly in information systems.
- a platform for extensive further research programs which are either directly related to this work or that flow from findings associated with this work.

1.4 Outline of Thesis

This thesis consists of eight chapters which are shown in Figure 1.1.

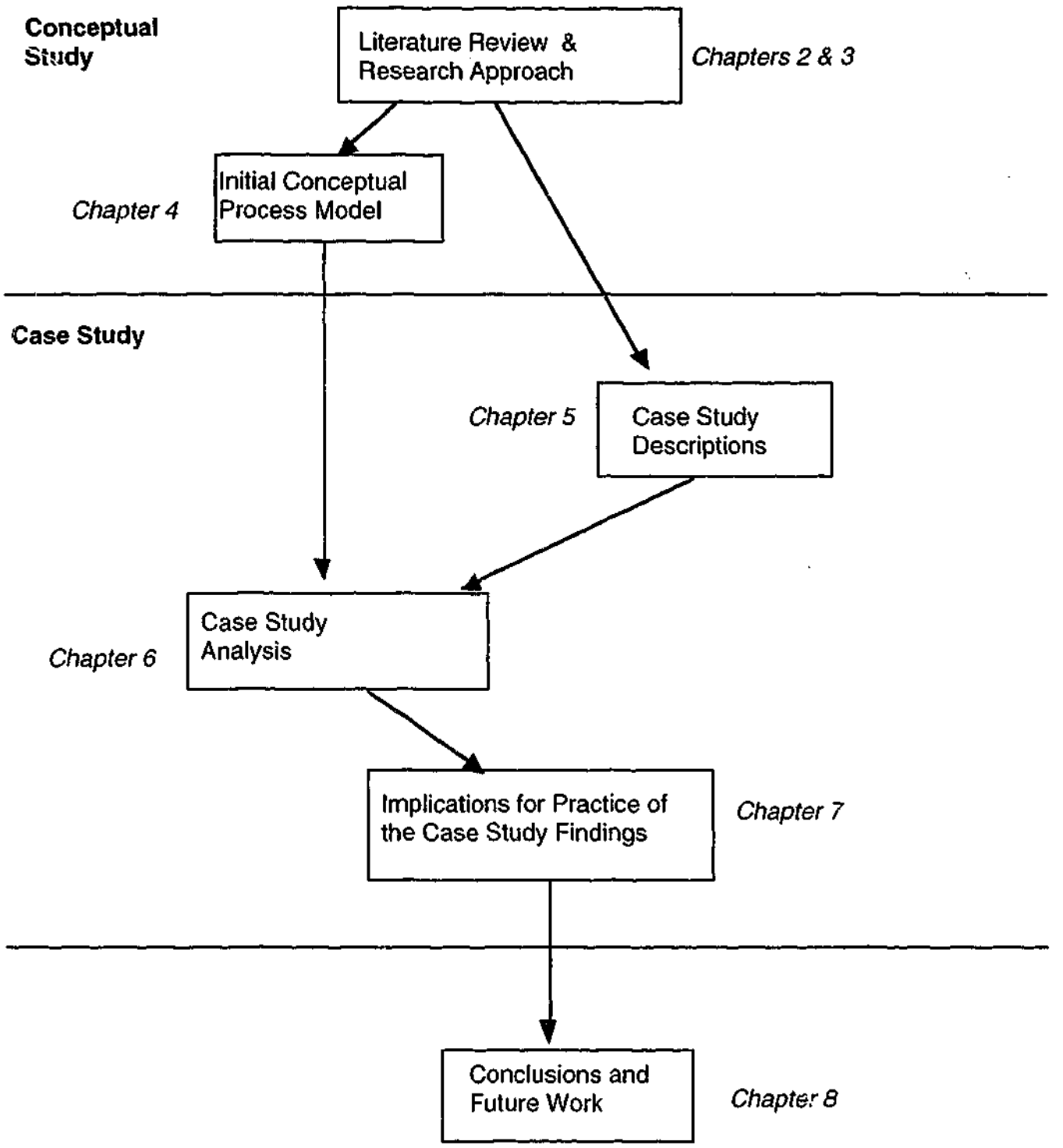


Figure 1.1 Structure of the Research Project

Chapter 2 provides an analytical review of the literature in two major areas: requirements engineering with particular reference to the processes of requirements engineering and the cognitive and social aspects of requirements engineering; object-oriented approaches to requirements engineering,

including object-oriented models and object-oriented development methodologies.

Chapter 3 presents the research approach for this project. It outlines the objectives of this research, the formal research questions and the unit of analysis used in the research project. The choice of research approach is discussed in the light of commonly accepted philosophies, paradigms and methods used in information systems research.

Chapter 4 describes the development of the initial conceptual process model.

Chapter 5 is a detailed description of the case studies undertaken in this project.

Chapter 6 presents the findings in terms of the empirical validation of the conceptual process model, the evolution of the conceptual process model through several versions to a final conceptual process model, and the cognitive and social aspects of object-oriented requirements addressed in the research questions.

Chapter 7 discusses the implications of the findings and the conceptual process model for practice and training.

Finally, Chapter 8 provides a summary discussion of the meaning of the findings in relation to the research objectives and research questions together with a discussion of proposed future research projects that follow from this project.

Chapter 2

Object-Oriented Requirements Engineering: Concepts and Approaches

2.1 Overview

The research focus of this project is the use of models and methods in object-oriented requirements engineering. This domain includes concepts from both the domain of requirements engineering in general and the domain of object-oriented systems development. This chapter provides a review of the literature within both these domains.

In addressing the object-oriented requirements engineering process this project takes a broad view which includes the technical or process model issues and the cognitive and social or human factor issues. In the following sections firstly, research concerning the nature of the requirements engineering process is outlined in section 2.2. Several definitions and frameworks of the requirements engineering process from the requirements engineering research literature are presented. This section presents the literature addressing the technical or

process model issues. Secondly, the cognitive aspects of requirements engineering within the context of general theories about problem-solving, reasoning and modelling are described and discussed in section 2.3. Thirdly, research studies of the requirements engineering process which address technical, cognitive and social aspects of requirements engineering are discussed. Finally, various approaches to object-oriented systems development are discussed in sections 2.5 and 2.6 with particular reference to the requirements engineering process.

The important issues which relate directly to this research project are:

- the features and characteristics of the requirements engineering process in general
- the cognitive and social factors which influence the conduct of requirements engineering
- the way the requirements engineering process is carried out in object-oriented system development and,
- the way object-oriented models are used in the object-oriented requirements specification process.

In addressing these issues, this research project proposes to define a specifically object-oriented view of the requirements engineering process in the form of a conceptual process model. The empirical validity of the model will be examined by conducting case studies of practising professional requirements engineers (or systems analysts) who are working with object-oriented system development methods and models.

2.2 The Requirements Engineering Process

Requirements engineering refers to the process or processes that occur early in information systems development in which the requirements for an

information system are determined and expressed as a requirements specification (Loucopoulos and Karakostas, 1995, Macaulay, 1996). Requirements engineering specifically addresses the problems associated with requirements specifications which may be ambiguous, incomplete or incorrect. Inadequacies in requirements specifications can result in errors and subsequently the costly maintenance or failure of software systems (Boehm, 1981, Lyytinen and Hirschheim, 1987, Jackson, 1997, OASIG, 1996).

A requirements specification document is, at least in the functional sense, an outcome of requirements engineering from which an information system is designed and implemented (Macaulay, 1996). A definition of a *requirement* from IEEE-Std.610 (IEEE-Std., 1990) is given as follows:

- A condition or capacity needed by a user to solve a problem or achieve an objective.
- A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
- A documented representation of a condition or capability as in 1 or 2.

There are several views of requirements engineering presented in the literature that are relevant to this project. Two definitions are given by Loucopoulos and Karakostas (1995)

> *"Requirements engineering deals with activities which attempt to understand the exact needs of the users of a software intensive system and to translate such needs into precise and unambiguous statements which will subsequently be used in the development of the system." [page vii] (Loucopoulos and Karakostas, 1995)*

> *"... the systematic process of developing requirements through an iterative co-operative process of analysing the problem, documenting the resulting observations in a variety of representation formats, and checking the accuracy of the understanding gained."*
> *[page 13] (Loucopoulos and Karakostas, 1995)*

The first definition emphasises the need for a precise, unambiguous specification from which to develop a system. The second definition emphasises the social process involved in eliciting, modelling and validating requirements.

Kotonya and Sommerville (1998) suggest that

> *"...there is no single [requirements engineering] process which is right for all organisations. Each organisation must develop its own process which is appropriate for the type of systems it develops, its organisational culture, and the level of experience and ability of the people involved in requirements engineering," p9.*

Macaulay (1996) similarly suggests nine different approaches which *"...vary depending on the interest of the groups from which they originate" p9.*

Although there is no commonly accepted definition of requirements engineering (Loucopoulos and Karakostas, 1995, Macaulay, 1996) the definition used in this thesis is (Dawson and Swatman, 1999):

> *"Requirements engineering is an iterative and collaborative process of elicitation, modelling, and validation of information system requirements which provides a specification which is the basis for the design and implementation of that information system." p355*

This definition places the requirements engineering process early in the system development cycle, although in many projects it may not necessarily be the very first phase. It may be that the requirements engineering process is triggered by preliminary investigations (Avison and Fitzgerald, 1995) or questionings of users and clients of the way activities are undertaken within their organisation (Checkland and Scholes, 1990). This definition also contains explicit reference to the collaboration needed between clients and analysts (Urquhart, 1998) and the need for feedback via iteration in the process of specifying requirements.

2.2.1 Defining and Modelling Requirements

Various formal, semi-formal and informal techniques are used for representing requirements (Pohl, 1994, Jarke et al., 1993) where the choice of representation technique depends on personal preference of the analyst and the current state of the specification (Pohl, 1994, Bubenko and Wangler, 1991). Informal representations (such as graphics, natural language and animations) are expressive and user-oriented (Flynn and Warhurst, 1994, Pohl, 1994, Weidenhaupt et al., 1998) whereas formal representations are semantically well defined and system-oriented (Pohl, 1994, Jarke et al., 1993, Swatman and Swatman, 1992, Greenspan et al., 1994). Semi-formal representations such as entity-relationship diagrams and data flow diagrams provide graphical visualisation but lack formal semantics (Pohl, 1994).

There are many methods used by successful systems analysts to produce robust requirements specifications: data-oriented methods (Chen, 1976, Martin and Odell, 1992); process-oriented methods (DeMarco, 1978, Gane and Sarson, 1978); object-oriented techniques and methods (Booch, 1994, Henderson-Sellers, 1997, Rumbaugh et al., 1991); or a combination of several methods. Most system development methods provide techniques for modelling requirements and for system decomposition as part of the structuring of system development activities. **Data-oriented** methods focus on modelling data requirements first

and then modelling processes as acting on the data in the models. The development of applications using this approach is data driven and is often based on entity-relationship (ER) modelling (Chen, 1976). **Process-oriented** methods focus on modelling process requirements and *then* building a data model to represent the data that the processes act upon. The development of applications is process-driven and is often based on Data Flow Diagrams (DFDs) (DeMarco, 1978, Gane and Sarson, 1978). **Object-oriented** methods model systems as sets of client and server objects in a problem domain and are based on various object-oriented modelling techniques and methods (Booch, 1994, Henderson-Sellers, 1997, Rumbaugh et al., 1991, Jacobson et al., 1992) which are described in section 2.5 below.

2.2.2 Existing Requirements Engineering Process Models and Frameworks

The material presented in this section regarding existing requirements engineering process models and frameworks is extended and dealt with in greater detail in Chapter 4 as the basis of the development of a conceptual process model of object-oriented requirements engineering.

Traditional systems development approaches or frameworks incorporate requirements specification as a separate phase within larger systems development methodologies. In these approaches, such as Structured Systems Analysis and Structured Systems Analysis and Design Method (SSADM) (Gane and Sarson, 1978, Downs et al., 1988), the process of requirements engineering produces a specification or requirements document which is the basis of design and implementation (Avison and Fitzgerald, 1995). Requirements are represented using standard models such as data flow diagrams, entity-relationship models and structured English.

Pohl (1993) proposes a framework suggesting three dimensions of the requirements engineering process. These three dimensions can be outlined as follows:

Specification Dimension involves the development of the specification from the "opaque" to the specific.
Representation Dimension which deals with the methods for representing the specification and include informal, semi-formal and formal languages.
Agreement Dimension describes the "common specification" or agreed specification which is based on the different viewpoints of the parties involved in developing the specification.

This view of requirements engineering is in harmony with the idea of information systems development being more than a purely technical undertaking. It incorporates the concept of an information system described by (Loucopoulos and Karakostas, 1995) as a "socio-technical system", that is, a system *"...that involve[s] computer-based components interacting with people and other technical system components in an organisational setting."*

Darke and Shanks (Darke and Shanks, 1996) propose a viewpoint framework based on four main elements. The *viewpoint agent* is a particular role or view of the problem domain adopted by one or more stakeholders. The *viewpoint development role* identifies the intended use of viewpoint development. The *viewpoint representation* is an informal, semi-formal or formal representation of a viewpoint associated with a particular agent and *the viewpoint development process* defines essential activities carried out in viewpoint development including representation. Viewpoint approaches to requirements engineering emphasise the collaborative nature of requirements definition and viewpoint development is the process of identifying and representing requirements from multiple stakeholder perspectives (Finkelstein et al., 1992, Darke and Shanks, 1996, Nuseibeh et al., 1994).

When considering the various existing frameworks of requirements engineering as the basis of the conceptual process model for this research project, the framework proposed by Loucopoulos and Karakostas (1995) is the most compatible with the definition of requirements engineering used in this thesis which was given above in section 2.2. This framework is shown in Figure 2.1. In this framework the requirements engineering process is broken down into three processes, elicitation, specification and validation, which deal with two external entities, the user and the problem domain. The purpose of elicitation is to obtain as much knowledge as possible about the problem in order to build a specification for the solution to the problem. Input comes from the user and existing information about the problem domain. Like the other two processes, elicitation does not end when the next process starts. Rather, each process relies on feedback from the other processes throughout the requirements lifecycle. The specification process provides models for validation by the user and against the original problem domain.

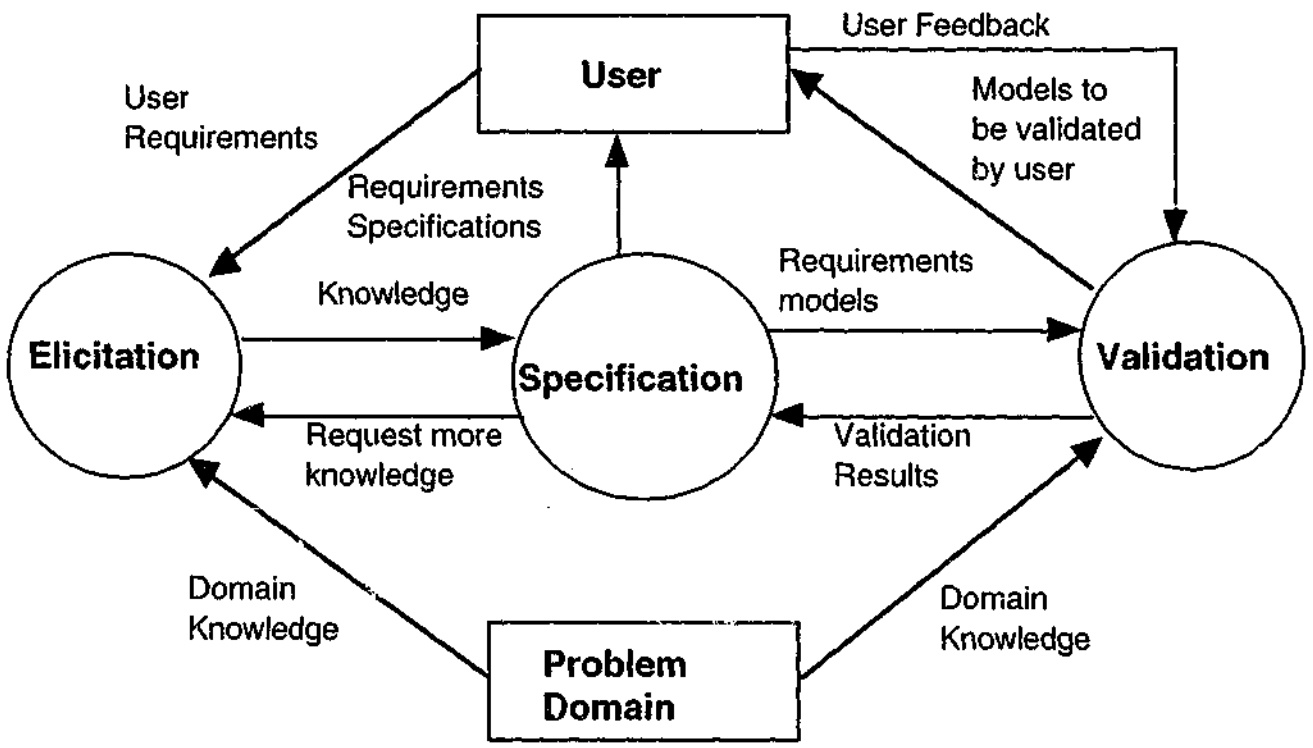


Figure 2.1: A framework for requirements engineering processes: Loucopoulos and Karakostas (1995) page 21

Loukopoulos and Karakostas (1995) suggest that the main outcomes from the elicitation process are the conceptual models which are domain-dependent models. The models become more software-oriented than problem domain-oriented as the requirements engineering process progresses.

2.3 Cognitive Processes in Requirements Engineering

Having examined the technical or process aspects of requirements engineering there is a need to describe the cognitive processes in the requirements engineering process. Based on Pohl's (1993) three dimensions of requirements engineering (described in section 2.2.2 above): specification, representation and agreement, it is proposed that the technical skills used to develop software systems that have traditionally been the focus of software engineering are not sufficient for understanding of the processes of elicitation, specification and validation of requirements. Underlying the social/organisational and cognitive aspects of the requirements engineering process are skills and concepts from other disciplines such as sociology, organisational science, and cognitive science (Urquhart, 1998, Hirschheim, 1985, Hirschheim and Klein, 1989, Checkland, 1981).

Successful requirements specification can be considered to be an art or a skill honed from a great deal of experience (Loucopoulos and Karakostas, 1995, Macaulay, 1996, Sommerville and Sawyer, 1997). In order to understand the art of requirements specification we need to understand how successful analysts operate within a requirements engineering environment.

2.3.1 Thinking and Problem Solving

Problem modelling for simulation or systems design is an important aspect of many design activities including architecture, engineering, modelling chemical processes, and economic modelling to name but a few (Schön, 1983, Khushalani

et al., 1994, Johnston, 1999, Galal, 1998). All of these design processes draw on the discipline of cognitive science or the study of human thinking and problem-solving. The cognitive design processes used in information technology-related disciplines, such as knowledge engineering, software engineering, database development and conceptual modelling for information systems, and the similarity and overlap of the cognitive processes involved in the design methods used in these disciplines, are of interest in understanding the modelling methods used for specifying requirements for information systems.

This section presents a background to problem modelling for information systems development. The aim is to investigate and gain a better understanding of problem modelling and problem solving in the context of information systems development by drawing on classical cognitive science definitions of problem solving and thinking. In sections 2.3.1, 2.3.2, 2.3.3 the way that general definitions of thinking, reasoning and modelling can be applied to information systems are highlighted and summarised in section 2.3.4.

Problem modelling in systems analysis is not necessarily aimed at modelling "problems" in the general sense (although the term *problem* is an often-used general term as it is in cognitive science) but is usually aimed at achieving some task. That is, if there is a perception of a task to be accomplished or a problem to be solved, a systems designer must first describe or model the problem or task in a way that facilitates progress towards accomplishing that task or solving that problem.

Problem solving can be defined as having three phases (Mayer, 1992, Polya, 1957):

- Understanding the problem in terms of what is known (givens) and what needs to be achieved (goal). i.e. describing *what* the task is.
- Planning a solution (using past experience where appropriate). i.e. reuse previous knowledge to describe *how* to achieve the task.
- Testing the result. i.e. validating and verifying the solution.

The investigation of thinking, problem solving and cognition has long been a major concern of cognitive psychologists, and now cognitive scientists are seeking to apply cognitive theory to information technology disciplines. General definitions of thinking and problems are a useful starting point for discussing problem modelling for information systems development.

Mayer (1992) defines human cognitive processes in terms of *problems* and *thinking*. He defines a *problem* as having the characteristics of *givens* and *goals*: and that " *... any definition of "problem" should consist of the three ideas that (1) the problem is presently in some [given] state, but (2) it is desired that it be in another [goal] state, and (3) there is no direct, obvious way to accomplish the change."*

Reitman (1965) categorises problems according to given and goal states as:

- Well-defined given state and well-defined goal state
- Well-defined given state and poorly defined goal state
- Poorly defined given state and well-defined goal state
- Poorly defined given state and poorly defined goal state

Greeno and Simons (1988) suggest a four-part topology of problems:

- Problems of transformation
- Problems of arrangement

- Problems of inducing structure
- Evaluation of deductive arguments

and that most problems include several of these aspects.

Mayer (1992) considers the terms *thinking, problem solving* and *cognition* to be equivalent, and although there is a serious lack of agreement among psychologists about the definition of thinking, Mayer suggests a compromise general definition that most psychologists might accept:

1. Thinking is *cognitive*, but is inferred from behaviour. It occurs internally, in the mind or cognitive system, and must be inferred indirectly.
2. Thinking is a *process* that involves some manipulation of a set of operations on knowledge in the cognitive system.
3. Thinking is *directed* and results in behaviour that "solves" a problem or is directed towards a solution.

In other words, thinking is what happens when a person solves a problem, that is, produces behaviour that moves the individual from the given state to the goal state.

2.3.2 Theories of Reasoning

Wilhelm Wundt (often called "the father of psychology") opened the first psychology laboratory in Leipzig in 1879. Wundt divided psychology into two parts: psychical processes such as physiological reflexes, perception etc. which could be observed; and higher psychical processes such as art, literature etc which could not be studied in a laboratory but could only be studied by looking at society as a whole.

The first experimental studies on human thought by Otto Selz in the early 1900s (Humphrey, 1963) used a method called *introspection*. A subject is given

a word and a question about the word, and asked to say the first word that enters his/her mind and describe the process which led to the answer. This is also known as a "thinking aloud protocol" and has been developed into *protocol analysis* (Ericsson and Simon, 1980) as a technique used in laboratory-based empirical research including in information systems research (Chaiyasut and Shanks, 1994, Guindon, 1990, Sutcliffe and Maiden, 1992).

Otto Selz used introspection to develop a theory independent of images and associations. His theory suggested that rather than a chain of associations, problem-solving involved completing a structural complex called *a unit of thought*. Selz's ideas foreshadowed modern cognitive psychology and included:

- the unit of thought is the directed association (as opposed to the undirected association)
- understanding a problem involves forming a structure
- solving a problem involves testing for conditions.

Further development of the discipline of psychology included behaviourism, gestalt and modern cognitive psychology. Modern cognitive psychology emerged in the so-called "cognitive revolution" (Gardner, 1985).

Associationism is based on the concept of learning by reinforcement. Associationist theory views problem-solving in terms of a problem situation (S) with associations to many possible responses (R_1, R_2, ... R_n) which vary in strength. Experiments using anagram solving and scenarios where animals escape from puzzle boxes by performing certain actions led to the idea of response learning based on trial and error. Thorndike (1898) formulated two laws of learning:

- the law of exercise where practice tends to strengthen an S-R link.

- the law of effect where unsuccessful responses tend to weaken an S-R link.

Mayer (1992) summarises this as *" ... associationists describe problem-solving as the trial and error application of a thinker's existing habit family hierarchy. In a new situation ... subjects try their most dominant response first, then their second strongest and so on."* Within associationist theory, mediational theory suggests a "train of thought" where a response (R) to a stimulus (S) is reached via a chain of covert (trial and error in the mind) s-r pairs:

$$S - r_1 - s_1 - r_2 - s_2 - \ldots , r_n - s_n - R$$

The theory of *transfer* (Thorndike, 1898) is still being investigated (Mayer, 1992). Transfer describes the effects of prior learning on new learning in terms of *positive transfer* - learning task A helps to learn task B and *negative transfer* - learning task A inhibits the learning of task B. Thorndike (1898) was interested in *specific* transfer - A and B identical, versus *general* transfer - A is a general cognitive skill helpful in learning B. For example, 'Does learning Latin help in learning French or does learning Latin provide good learning habits for thinking in general?'. The concept of transfer may be relevant to the idea of professional systems analysts being able to relate the learning of one technique, tool or modelling notation to another.

Gestalt theory views problem-solving as looking at a problem in a new way or restructuring or reorganising a problem using *insight* resulting in a structural understanding of how all the parts of the problem fit together to satisfy the goal. Gestalt theory suggests that there are two types of thinking: productive thinking which produces a new organisation using insight; and reproductive thinking which reproduces old behaviours by trial and error.

Polya (1957) suggested four steps in (mathematical) problem-solving:

- Understand the problem - information gathering and questioning

- Devise a Plan - find a method of solution based on past experience [either backwards from goals previously achieved or forwards by restating the givens]
- Carry out the Plan - plan and check each step of the solution
- Look back - check the result.

There have also been experiments (Duncker, 1945, Luchins, 1942, Bartlett, 1958) which identify "functional fixedness" where past experience inhibits successful problem-solving.

Inductive reasoning or hypothesis testing is based on concept learning. This is related to the idea of classification where one tries to determine whether a new instance belongs to a specific concept-class. For example, 'Is a four-legged furry animal a cat or a dog?'. Inductive learning is based on inducing rules or hypotheses and testing them. If the rule can predict class membership for any instance then the hypothesis is retained otherwise a new hypothesis is generated. Inductive reasoning treats thinking as hypothesis testing. This type of reasoning in terms of identifying members of classes is typically used in entity-relationship and object-class modelling for system specification.

Bruner et al (1956) suggest three types of classification rules:

1. Single-value concepts - based on a single attribute
2. Conjunctive concepts - based on two concurrent attributes
3. Disjunctive concepts - based on an instance having a specific attribute in one case or a different attribute in another case.

An alternative to hypothesis testing models involves averaging all instances of a category into a prototypical instance or *abstraction of prototype*.

Klahr and Dunbar (1988) set up experiments to discover how subjects learnt to operate a robot tank. Subjects were taught how to use a set of ten keypad instructions and then asked to work out how the RPT key worked. They could enter and run as many programs as they wanted until they worked out what the RPT key did. Subjects were also asked to think aloud by stating their current hypothesis and saying what they were thinking as they tried different keys and the robot responded. Analysis showed that subjects made two kinds of selections: select a hypothesis; and select an experiment to test the hypothesis. Subjects often failed to seek negative evidence. Rather they used positive test strategies (Klahr and Dunbar, 1988). That is, `This is what should happen'. In 60% of experiments, hypotheses were falsified by subsequent testing but approximately half the subjects retained their hypothesis and retested it. Successful strategies included testing alternative hypotheses of different types and designing experiments to explain unexpected results. Unsuccessful or slower strategies involved focussing on a single hypothesis of one type and searching for confirmation of a hypothesis. Klahr and Dunbar (1988) argue that positive test strategies can lead to effective reasoning in cases where the probability of obtaining confirmation is low and therefore humans can be successful scientific reasoners.

Deductive reasoning or drawing logical conclusions is based on *syllogisms*. A syllogism consists of two premises and a conclusion. There are three types of syllogisms (Mayer, 1992):

1. Categorical syllogisms - all A are B; all B are C; therefore, all A are C.
2. Linear syllogisms - A is greater than B; B is greater than C; therefore, A is greater than C.
3. Conditional syllogisms - if p then q; p is true; therefore q is true.

That is, given some premises that are accepted as true, a person should be able to draw a logical conclusion. *Categorical reasoning* is based on relationships

between categories, or sets of things and is often represented using Venn diagrams. For example,

All A are B is the universal affirmative (UA)
No A are B is the universal negative (UN)
Some A are B is the particular affirmative (PA)
Some A are not B is the particular negative (PN)

Most propositions are ambiguous. Although "All A are B" does not necessarily imply "All B are A" it does not rule it out. Also "some" means "at least one and possibly all". Some of these concepts relate to cardinality in entity-relationship modelling and multiplicity in object modelling, that is, specifying how many instances of one class or entity-set may relate to an instance of an associated class or entity-set.

Human errors in syllogistic reasoning have two basic theoretical explanations:

1. Encoding theories - incomplete or incorrect interpretation of premises and/or conclusions.
2. Processing theories - incomplete or nonlogical processing of premises.

Linear reasoning is based on the three term series task (Mayer, 1992). For example,

Bill is shorter than Allen.
Bill is taller than Charles.
Therefore, Allen is the tallest of the three.

Two issues are of interest in linear reasoning. Firstly, do subjects use a visual strategy or a verbal strategy? Secondly, do subjects integrate the two premises into a single representation or do they encode individual facts? A visual

strategy might relate to the use of graphical models in an analysis or design modelling process.

Conditional reasoning is based on two premises and a conclusion of the form: "if p then q". That is

Premise 1: If p then q, where p is an *antecedent* condition and q is a *consequent* condition leads to

Premise 2: Affirmation of antecedent - p is true OR
Denial of antecedent - p is false OR
Affirmation of consequence - q is true OR
Denial of consequence - q is false.

For example,

If there is a solar eclipse, then the streets will be dark.
There is a solar eclipse.
Therefore the streets are dark.

Studies (Staudenmayer, 1975, Staudenmayer and Bourne, 1978, Taplin and Staudenmayer, 1973) have shown that human errors in conditional reasoning are due to subjects misrepresenting the first premise rather than errors in logic. Most often subjects interpret "if p then q" to include "if q then p". This is called a *biconditional relation*. It is also known as *abduction* or *abductive inference* in knowledge-based systems (KBS). Menzies and Compton (1995) give a definition as follows:

> *"Consider a system with two facts a, b and a rule R_1: if a -> b. Deduction is the inference from a to b. Induction is the process of learning R_1 given examples of a and b occurring together. Abduction*

is inferring a, given b. Abduction is not a certain inference and its results must be checked by an inference assessment operator."

Abduction is used in expert systems development as a heuristic which may help reduce the search space by generating reasonable hypotheses which can then be used with deduction. Abduction is sometimes referred to as *"...reasoning from observed facts to the best possible explanation"* (Reggia, 1985). Jackson (Jackson, 1990) calls abduction *"... reasoning from observed effects to possible causes."* Giarratano and Riley (1989) describe abduction as *"... a fallacious argument ... [which] ... is not a valid deductive argument"* and cannot provide true conclusions. For example (Giarratano and Riley, 1989),

If x is an elephant THEN x is an animal.
If x is an animal THEN x is a mammal.

Can we conclude that Clyde is an elephant if we know that Clyde is a mammal? In the real world we cannot but in a *closed world* with only the two rules above we can. So, adding another rule, "If x is a dog THEN x is an animal" means that Clyde is either an elephant or a dog in our closed world.

Analogical Reasoning is based on analogues, models and examples and *"... pervades all our thinking."* (Polya, 1957). Mayer (1992) defines analogical reasoning as abstracting a solution strategy from one problem and relating that information to a new problem where the original domain is called the *base* domain and the domain to be explained is called the *target* domain. A critical skill in the cognitive processes required in conceptual modelling and conceptual reuse based on experience is the ability to apply the principle of abstraction. Abstraction is a key aspect of analogical reasoning.

Two analogical problems may have a *surface similarity* where the two problems share common characteristics which may not be related to the

solution and/or a *structural similarity* where the relations of objects in one problem correspond to the relations of objects in the other problem. If solving one problem helps to solve another problem we can say there has been an *analogical transfer* of problem solving strategy. Mayer (1992) goes on to show that it is *how* the analogies are presented which influences the positive or negative outcome of analogical transfer.

Mayer describes three conditions for successful analogical transfer:

1. *Recognition* - in which a problem solver identifies a potential analog (or base) from which to reason,
2. *Abstraction* - in which a problem solver abstracts a general structure or principle or procedure from the base, and
3. *Mapping* - in which a problem solver applies that knowledge to the target.

Mayer divides analogical reasoning into thinking using analogues, thinking using models and thinking using examples. An analog has structural similarity but not surface similarity with a target problem.

Subjects' **recognition** of structural similarity seemed to depend on hints. Subjects were given three stories, two unrelated to the problem and the other structurally similar. If subjects were told "one of the stories you read before will give you a hint for a solution to the problem" there was a 92 percent success rate against a 20 percent success rate without the hint. Knowing a solution plan for an analogous problem is not very useful unless you realise that the problem is analogous to the one you are working on. Studies in this area (Gentner, 1989, Gentner, 1983, Gick and Holyoak, 1980, Holyoak and Koh, 1987) show that experience and structural similarity of problems are critical in analogical transfer.

Encouraging subjects to **abstract solutions** from analogues has been the subject of several studies. Gick and Holyoak (1980) propose a *schema induction theory*

which suggests that it is easier to induce a general schema from experiences with structurally similar problems in different domains than from a single problem.

Mapping knowledge from a base analogy to a target analogy depends on the overall analogy transfer, that is, recognition of an analogous problem, then the abstraction of useful information that can be used to solve another problem. Holyoak and Koh (1987) propose that the more surface features between two analogues the more likely that subjects will actually map the knowledge from one problem to the other. Recognition of structural similarity or some form of analogical transfer may be a factor for experienced analysts when approaching new problems or systems development projects.

2.3.3 Cognitive Modelling

One of the major focus areas of this research project is the use of models, particularly object-oriented models, in the requirements engineering process. Mayer describes a **model** of a system as a system *"[which] includes the essential parts of the system as well as the cause-and-effect relations between a change in status of one part and a change in status of another part"*. He uses Gentner's (Gentner, 1983, Gentner, 1989) *structure-mapping theory* where knowledge about one system (the base) is used to reason about another system (the target). Gentner's system consists of *objects* with *attributes* and *relations* between objects (similar to object-oriented modelling methods). Studies based on using a water-flow model and a moving-crowd model to understand and reason about an electrical system showed that subjects could reason about electrical circuits based on knowledge of a water-flow model or a moving-crowd model. Mayer and Gallini (1990) also showed in a series of experiments concerning reasoning about models for explaining how radar and pumping systems work, that adding pictorial models to textual descriptions improved problem solving performance by an average of over 60 percent. Among novice subjects the

model-illustration group produced almost twice as many useful answers to the questions than the incomplete-illustration or no-illustration groups. This is one aspect of modelling investigated in this research project.

The term **mental models** are defined in cognitive psychology (Johnson-Laird, 1983, Craik, 1943, Norman, 1983) as either analogical representations or a combination of analogical and propositional representations. Preece (1994) suggests that *"A mental model represents the relative position of a set of objects in an analogical manner that parallels the structure of the state of objects in the world."* In the field of human computer interaction (HCI) a mental model has been defined as *" ... the model people have of themselves, others, the environment, and the things with which they interact. People form mental models through experience, training and instruction."* (Norman, 1988, p 17). Mental models are used extensively in HCI to explain dynamic aspects of cognitive activity and it is suggested that people build mental models of the world in order to make predictions about an external event before carrying out an action (Preece, 1994). Mental models, as specific examples of analogical models, may be used by analysts when developing requirements or design models.

2.3.4 Summary of cognitive processes in requirements engineering

The aim of this section was to gain a better understanding of problem modelling and problem solving in the context of information systems development by drawing on classical cognitive science definitions of problem solving and thinking. The examination of the literature highlighted the following aspects of modelling and problem solving for information systems development:

- introspection and association are methods used in *protocol analysis* (Ericsson and Simon, 1980), a technique used in laboratory-based empirical

information systems research (Chaiyasut and Shanks, 1994, Guindon, 1990, Sutcliffe and Maiden, 1992)

- the theory of transfer (Thorndike, 1898) may be relevant to the idea of professional systems analysts being able to relate the learning of one technique, tool or modelling notation to another.
- inductive reasoning or concept learning where members of specific concept-classes are identified is similar to techniques used in entity-relationship and object-class modelling for system specification.
- deductive reasoning involving concepts of cardinality or multiplicity, that is, specifying how many instances of one class or entity-set may relate to an instance of an associated class or entity-set is evident in entity-relationship modelling and object modelling
- visual strategies used in linear reasoning may be the basis of graphical models in an analysis or design modelling process.
- explicit conditional reasoning based on two premises and a conclusion and the variation of conditional reasoning known as abduction used in expert systems development may assist system modellers in generating reasonable hypotheses which can then be used to reason about the logic of the models that they build.
- a critical skill in the cognitive processes required in conceptual modelling and conceptual reuse based on experience is the ability to apply the principle of abstraction.
- recognition of structural similarity or some form of analogical transfer may be a factor for experienced analysts when approaching new problems or systems development projects.
- the definition of general conceptual models as consisting of *objects* with *attributes* and *relations* between objects maps directly to object-oriented modelling approaches.

- mental modelling may be used by analysts when developing requirements or design models to test models privately before developing more concrete paper-based models.

2.4 Studies of the Requirements Engineering Process

Having examined and described problem solving and modelling for information systems within the context of traditional cognitive science definitions of problem solving and reasoning this section presents research in information systems specification and modelling based on cognitive and social factors.

Several studies (Chaiyasut and Shanks, 1994, Guindon, 1990, Loucopoulos and Karakostas, 1995, Sutcliffe and Maiden, 1992, Vitalari and Dickson, 1983) have looked at how analysts apply general problem solving and reasoning skills to the process of requirements engineering. Some factors affecting requirements specification as a problem solving activity are as follows (Loucopoulos and Karakostas, 1995):

- Analysis problems are ill-defined and constantly changing as the organisational context changes and more information about user requirements is gathered
- Requirements exist in organisational contexts and may be conflicting from differing viewpoints
- The process of analysis is a cognitive activity, requiring understanding of an abstract problem, and development of a logical and internally consistent set of specifications.

Some characteristics of successful analysts include (Loucopoulos and Karakostas, 1995):

- have acquired interdisciplinary knowledge and skill.
- are flexible and ready to incorporate changes in the technology and to participate with users in different ways.
- have highly developed interpersonal and organisational skills which facilitate the problem solving process.
- use information from the problem domain to classify problems and relate them to previous experience.
- often start solving a problem by forming a mental model which is progressively refined as further information is obtained.
- develop hypotheses about possible solutions which are discarded if they are refuted and retained and retested otherwise.
- almost always summarise at intervals during client/developer interviews in order to verify their findings.

Several studies have been undertaken which investigate the cognitive processes used by novices and experts in systems analysis and design (Sutcliffe and Maiden, 1992, Guindon, 1990, Chaiyasut and Shanks, 1994, Morris et al., 1996). The method used most often in these studies is *protocol analysis* (Ericsson and Simon, 1980) which is based on verbal reporting of activities or thinking aloud protocols. Subjects are asked to describe their thought processes while undertaking some analysis or design task and these verbal reports are recorded on audio or videotape for later analysis.

Sutcliffe and Maiden (1992) used protocol analysis to investigate the cognitive problem-solving behaviour of novice systems analysts performing a requirements analysis task. The main aim of the study was to model the behaviour of novice analysts in order to form the basis for a diagnostic module of an intelligent tutor for a CASE tool. In this instance a CASE tool is viewed as both a problem solving tool and a learning tool where the diagnostic module

attempts to determine a user's knowledge state (given state) and a didactic module aims to lead the user to a required knowledge state (goal state).

The study used 13 MSc students in a Business Systems Analysis course who were asked to develop a specification for a delivery scheduling system using Structured Systems Analysis (SSA) (Downs et al., 1988) methods and data flow diagrams (DFDs) (DeMarco, 1978). Subjects were given a 400-word description of a manual scheduling system.

> *"Subjects were requested to think aloud and their verbal protocols [concurrent protocols] were recorded on audio tape ... [After some practice at thinking aloud] ...each subject was requested to develop a specification of a computerized system using the SSA and data flow diagramming techniques taught on the MSc course ... Subjects were given 35 minutes to develop a specification ... Upon completion of the task a 10 minute retrospective protocol elicited further details of reasoning strategy and behaviour ... A marking scheme was created from solutions developed by three experienced analysts who had considerable knowledge of the delivery scheduling problem and SSA techniques. ...Subjects received a score if a component was included in the resulting data flow diagram or if the subject verbally stated that the component was to be included in the system [giving a completeness score as a percentage]."*

Subjects' behaviour was analysed using six major categories divided into mental and non-mental behaviours:

Mental:

- Recognise goal - statement of the high-level problem goals used in structuring the problem space.
- Assertion - the verbalisation of a belief or statement of fact.

- Reasoning - the verbalisation of the creation, development and testing of hypotheses.
- Planning - SSA plans and general plans.

Non-mental:

- Information Acquisition - from problem text or elsewhere.
- Conceptual Modelling - construction of specification using DFDs.

Since all the behaviours listed above could be called "mental" or cognitive, perhaps a more useful definition of the division could be *cognitive* behaviours and *task-directed cognitive* behaviours as used by Ericsson and Simon (1980). Planning was differentiated from goal recognition as " *...goals state what the system must achieve whilst plans state how the subject develops a specification to meet those system goals.*"

Inductive reasoning in this group was categorised as a process of generate, develop, test, confirm, modify or discard. Analysis of "mental" behaviours showed a decline in information gathering before an increase in goal recognition. Use of model-based or analogous reasoning by some subjects suggested that the use of conceptual modelling techniques may improve analytical performance or greater experience with conceptual modelling may promote model-based reasoning.

Subjects' overall performance was poor, averaging only 11.4% of the expert score. Solutions were incomplete rather than erroneous. Analysis of protocols suggested that subjects were unable to recognise or infer the processes and data store accesses included in the expert analysts' solutions.

Overall, performance in analysis was not linked to any one factor although reasoning was correlated with success. Poor performance was ascribed to failure

to scope the problem, poor formation of a conceptual model of the problem domain or insufficient testing of hypotheses. Good performance was indicated by well-formed conceptual models and good reasoning/testing abilities.

This leads Sutcliffe and Maiden (1992) to suggest that *"Formalisation of conceptual models in tractable notations may be one of the more important improvements which structured methods have made in supporting the analytical reasoning process."*

Guindon (1990) used protocol analysis to study how designers exploit knowledge in software systems design. Of interest in this study is the examination of how the designers defined and clarified the problem domain from informal system requirements to a specification suitable for producing a final systems design. A main premise in the study is that high level software design is characterised by incompletely specified requirements, no predetermined solution path, and by integration of multiple domains of knowledge at various levels of abstraction and further, *"[i]ncomplete and ambiguous system requirements (or goals) are intrinsic to system design ... [d]esign tasks involve extensive problem structuring. Problem structuring is the process of uncovering missing information and using it to define a problem space."*.

Guindon (1990) also found that designers rely heavily on problem domain scenario simulations throughout solution development and that "...*these simulations trigger the inferences of new requirements and complete the requirement specification."* There were three designers participating in the study who were all considered to be extremely competent and experienced by their peers and supervisors. The problem statement was based on the Lift Control Problem, a standard problem in the areas of software specification and software requirements research. The goal was to design the control logic to move N lifts between M floors, given several rules. The specification was

considered to be informal, and therefore incomplete and ambiguous, as well as knowledge-rich over several domains. Thinking aloud reports were collected from the designers over a period of two hours and participants were videotaped. Notes, diagrams and transcripts were time-stamped at regular intervals. When describing and clarifying the problem domain the designers performed scenario simulations in the problem domain *"... a subset of the real world with which a computer system is concerned (but not the design solution describing the computer system itself.)"*. The study identified five main uses for the scenario simulations:

- Understanding the given requirements.
- Understanding the inferred requirements.
- Developing solutions with scenarios.
- Unplanned discovery of new requirements.
- Unplanned discovery of partial solutions.

The study describes an activity observed in this study as the *"elaboration of the requirements - any activity whose purpose is to decrease the incompleteness and ambiguity of the informal requirements specification"* including inferred constraints not explicit in the original specification which can be deduced from knowledge of the problem domain. The inferred and added requirements reduce the ambiguity and incompleteness in the original specification and also reduce the range of possible solutions.

Morris et al (1996) conducted an empirical study to test a set of hypotheses which investigated the use of procedural and object-oriented techniques by experienced and novice systems analysts. Morris et al were interested in whether claims that object-oriented methodologies provide more complete solutions with reduced complexity compared to traditional process oriented methods were supported and whether experience in one method facilitates the learning and application of another method.

Morris et al propose that the research method they use differs from previous research in that it uses an empirical partial factorial design; has a larger sample size (71 subjects); and is based on a previously published cognitive processing model. They believe this is the first time that a subjective mental workload (SMW) (common in the human factors literature) has been examined in an IS context. Six hypotheses were set up for this study:

[H1A.] For novice systems analysts, the subjective mental workload (SMW) using modified Coad and Yourdon Object-Oriented Analysis (OOA) will be lower than the SMW using DFDs.
[H1B.] For procedurally experienced systems analysts, the SMW using DFDs will be lower than the SMW using modified Coad and Yourdon OOA.
[H2.] For novices, analysts applying modified Coad and Yourdon OOA followed by DFDs will differ in their ratings of SMW for each technique from analysts applying DFDs followed by modified Coad and Yourdon OOA.
[H3A.] For experienced systems analysts, time to solution using Coad and Yourdon OOA will be higher than time to solution using DFDs.
[H3B.] For novice systems analysts, time to solution using DFDs will be higher than time to solution using modified Coad and Yourdon OOA.
[H4.] Across all subjects, there will be a significant positive correlation between mental workload and time to solution.

The research design for this study was a 2X2X2 repeated measures, partial factorial design, with experience and method order as the between subjects variables and the analysis method as the within subjects variable. The method order received by subjects was counterbalanced to control for learning. Results showed that H1B and H2 were supported and H4 was partially supported. Morris et al suggest that *"[t]he theoretical model which was developed and tested in this study suggested that there would be an interaction between experience level and analysis method on SMW, and time to solution ... Instead, there appeared to be a main effect for method across all levels of*

experience. Specifically, the use of DFDs consistently resulted in lower SMW than OOA." and *" ...the SMW when using OOA is always greater than the SMW when using DFDs, regardless of the order in which it is performed.".* Further attitudinal measures *" ... collected for potential post hoc analyses ...are interesting because, on the surface, they appear contradictory to the results obtained from testing hypotheses 1A and 1B. However, it is possible that while novice subjects preferred OOA, the implementation of the method required greater cognitive effort."*

Morris et al acknowledge several limitations in their study. Specifically, the quantitative nature of the study and the use of SMW rather than protocol analysis (which generally gives a more qualitative view) limited the ability to capture the true mental models of the subjects. Other major limitations, it could be argued, are the assumption that final year students can be classified as "experienced analysts", the use of Coad and Yourdon's OOA methodology rather than one of the more commonly used methods such as Booch's (Booch, 1994), Rumbaugh et al's (Rumbaugh et al., 1991) or Jacobson's (Jacobson et al., 1992) and the fact that the OOA method was "operationalized" or modified in order to control the complexity differences between DFD and OOA modelling. Other limitations identified by Morris et al include the motivation of subjects, inadequate training on alternative techniques, and a concentration on internal validity at the possible expense of external validity.

Some recent work has addressed the idea that system design and requirements engineering is opportunistic and design and information gathering activities are triggered by information as it is gathered. Carroll and Swatman (1997) have argued the requirements engineering process is neither linear nor cyclic process of specification evolution but rather it is complex, ill-structured and often opportunistic. This view concurs with the work of Khushalani et al (1994) who showed, through an extensive and detailed protocol analysis-based study, that

"design" in a software context is opportunistic and insight-driven rather than incremental and evolutionary.

Rasch and Tosi (1992) set up a study to investigate factors affecting software developers' performance. This study was a psychological study based on anonymous questionnaires and existing personnel information. The integrated research model incorporated elements of expectancy theory, goal-setting theory and individual characteristics and analysed 335 systems developers in three large organisations. Individual participants were asked to fill out a questionnaire for self-assessment and companies were asked to provide internal company information. Research variables were performance, effort, role ambiguity, achievement needs, locus of control, self-esteem, goal specificity, goal difficulty, intellectual ability and quality of education. Results indicated that *" ... a software developer's ability and individual need for achievement were the two strongest factors determining individual performance."* ...and ..." *...that a software developer's motivation or level of effort was another important factor in determining individual performance."*

When considering requirements specification from the users' point of view there is some evidence that untrained users have difficulty in understanding the standard data models and object/class models that many professional analysts use during requirements and system modelling (Vessey and Conger, 1994). Various approaches to representing requirements in a manner that is understandable to untrained users have been suggested, often based on use cases and scenarios. Jacobson designed use case models (Jacobson et al., 1992, Jacobson, 1995) as a *"...first system model ...[which must] ...be comprehensible by people both inside the development organisation ... and outside... Object models are too complex for this purpose"*, but he also suggests that use cases serve many other purposes as well. These other purposes seem to contradict the purpose that use cases were originally designed for as the following quote reveals:

"Use cases serve several important purposes. Among other things, use cases are the basis for:

1. *defining functional requirements*
2. *deriving objects*
3. *allocating functionality to objects*
4. *defining object interaction and object interfaces*
5. *designing the user interface*
6. *performing integration testing*
7. *determining development increments*
8. *composing user documentation and manuals.*

They also help define the system and control development by serving as the vehicle for:

1. *capturing and tracing requirements*
2. *envisioning an application*
3. *communicating with end users and customers*
4. *delimiting a system*
5. *estimating project size and required resources*
6. *defining database-access patterns*
7. *dimensioning processor capacity " (Jacobson and Christerson, 1995) [page 15-19]*

Some recent research has investigated scenarios which are described by Kotonya and Sommerville (1998) as *"...stories which explain how the system is used."* Weidenhaupt et al (1998) provide an exploratory study of the use of scenarios in current practice. They found that *"...while many companies express interest in Jacobson's use case approach, actual scenario usage often falls outside what is described in textbooks and standard methodologies."*. They also found different forms of scenarios such as narrative text, structured text, diagrammatic notations, images, animations and simulations.

Studies of requirements engineering and systems development from a social and organisational perspective (Hirschheim and Klein, 1992, Checkland and Scholes, 1990) are also important to the foundations of this project. Based on the following definition (Dawson et al., 1995)[page 3],

> *"An information system is a collection of people, machines, software, data and procedures which must be designed, implemented and managed to fulfil a given proposal within an organisational environment."*

information system development can be considered as an outward looking, holistic process where systems (computerised and otherwise) are analysed, designed and developed within an organisational and social context. These ideas were explicitly explored and presented by Galal and McDonnell (Galal and McDonnell, 1998). They suggest the following:

- Firstly, requirements themselves are subjective (and political) and therefore must be interpreted by the professional.
- Secondly, both requirements specification and requirements validation are social processes. Both these processes involve interaction between a professional analyst and a client where problems and ideas are explored and solutions and actions are agreed upon.
- Thirdly, for this interaction to take place in a meaningful and useful way the language used is very important. In this research project, the language is the modelling representation selected and its notations.

Social and organisational aspects of requirements engineering including communication skills and analyst-client relationships have been a consistent issue in Information Systems (IS) literature for over 20 years (Urquhart, 1998). Macaulay (1996) found the following skills (amongst other skills) were

considered necessary for a requirements engineer from a survey of 32 companies: interviewing skills, groupwork skills, facilitation skills, negotiation skills and analytical skills. Similarly, Darke and Shanks (1997) suggest that defining requirements is a collaborative effort which relies on communication and interaction between stakeholder groups. Urquhart (Urquhart, 1998) found that analysts brought strong individual influences to the requirements definition process and that in individual cases *" ... it is possible to see how a myriad of situational influences contribute to what is discussed and in turn shape initial requirements."*

2.5 The Object Oriented Paradigm

The use of object-oriented approaches to system modelling and design arose out of a perceived need to address the problems of increasing system complexity and the need to reuse components from previously designed systems. The object-oriented paradigm combines data and procedural abstraction by defining objects and their behaviour. Complexity is said to be reduced because the concept of an object remains the same throughout the development process from analysis to implementation and flow of control is modelled as interactions between objects.

The move towards the use of object-oriented methods for information system development has led to a need for the development of object-oriented approaches to requirements engineering. Object-oriented methodologies can generally be thought of as having evolved from programming languages and design modelling (Graham, 1994). Most of the commonly used texts emphasise object-oriented design (Booch, 1991, Rumbaugh et al., 1991, Meyer, 1988). Object-oriented analysis and requirements specification is a relatively new addition to most object-oriented development methodologies and many methodologies take a fairly traditional, structured approach to the early phases of requirements

specification with the emphasis on deliverables (Booch, 1994, Graham et al., 1997, Jacobson et al., 1999).

Object-oriented models and methodologies (Booch, 1994, Coad and Yourdon, 1991, Henderson-Sellers and Edwards, 1994, Meyer, 1988) are claimed to provide a more natural way of specifying, designing and implementing information systems based on features which include:

- a consistent underlying representation of an identified object throughout the development process
- the encapsulation of static or descriptive characteristics together with the dynamic or behavioural characteristics of an object
- the ability to model complex systems by reusing objects and object components from previously designed systems
- the incorporation of high-level data abstraction facilities including inheritance and polymorphism

Object-oriented approaches to system development are considered to be a new way of thinking about problem solving and developing solutions which view a system as a collection of objects where each object is responsible for a specific task. The characteristics of object-oriented systems are (Budd, 1997, Henderson-Sellers, 1997, Graham, 1994):

- processing and computation proceeds by interaction between objects
- an object is an encapsulation of state (data values) and behaviour (methods)
- behaviour is dictated by an object's class. ie all object instances of the same class behave in a similar way (invoke the same methods)
- an object will exhibit behaviour by invoking a method in response to a message

- objects and classes extend the concept of abstraction by adding inheritance. ie classes can be organised into a hierarchy (tree). Data and behaviour of classes higher in the tree can be accessed by classes lower in the tree.
- by reducing interdependency among software components object-oriented approaches permit the development of reusable software ie components can be created and tested as independent units
- reusable software components allow a higher level of abstraction, ie we can define and manipulate objects simply in terms of the messages they understand and a description of the tasks that they perform, ignoring implementation details

2.6 A Review of Major Object-oriented Methods

Object-oriented systems development life cycles are usually based on non-linear cycles such as the spiral model (Boehm, 1988) or the fountain model (Henderson-Sellers, 1997). Henderson-Sellers (1997) reflects the conventional wisdom of the object-oriented community in stating that object-oriented *analysis* provides an accurate picture of a real world situation, object-oriented *design* supports good software engineering design and the goal of a good object-oriented method is the "seamless" transition between the analysis and design phase. Further, it is generally agreed within the object-oriented community (Coad and Yourdon, 1991, Meyer, 1988, Henderson-Sellers, 1997, Jacobson et al., 1992) that one of the strengths of object-oriented methods is that complexity is reduced because the concept of an object remains the same throughout the development process from analysis to implementation, and flow of control is modelled as interactions between objects.

Taylor (1992) provides a simplified view of the object-oriented life cycle based on this concept which uses the following phases:

- Object Descriptions corresponding to analysis
- Object Structures corresponding to design
- Object Code corresponding to implementation
- Object Testing corresponding to testing
- Object Extensions corresponding to maintenance.

Object-oriented analysis involves the specification of user requirements and the specification of system structure and function. It is independent of implementation and includes business analysis and synthesis in terms of:

- abstracting user requirements
- identifying key domain objects
- assembling objects into structures for design.

Graham, (1994) suggests that there are three aspects which often lead to three models of a object-oriented system:

- Data models - which model objects and their structure using entity-relationship (ER) (Chen, 1976) style diagrams
- Process models - which define the system architecture and the processes of interaction using dataflow (DFD) type diagrams (Gane and Sarson, 1978, DeMarco, 1978)
- Control models - which define dynamics of systems using state transition (STD) type diagrams

Graham, (1994) also divides object-oriented methods into three categories according to the following categorisation of model types:

- Ternary models - which have three separate notations for data, process and control eg OMT, (Rumbaugh et al., 1991), Ptech, (Martin and Odell, 1992)
- Unary models - which have one notation since objects inherently combine data and processes eg RDD (Wirfs-Brock et al., 1990), OOA (Coad and Yourdon, 1991)
- Hybrid models - which encompass characteristics of both ternary and unary models eg MOSES, (Henderson-Sellers and Edwards, 1994), SOMA, (Graham, 1994).

Henderson-Sellers (1997) uses the term "hybrid" in a different way. He describes a "pure" object-oriented methodology as encompassing object-oriented analysis, object-oriented design and object-oriented programming or O-O-O. Two hybrid methodologies are defined as comprising functional analysis, object-oriented design and object-oriented programming (F-O-O) and object-oriented analysis, object-oriented design and functional (procedural) programming (O-O-F).

The following overview is based on the original sources together with some insights from published surveys (Graham, 1994, Henderson-Sellers, 1997, Henderson-Sellers and Edwards, 1994, Loosley et al., 1994, Fichman and Kemerer, 1992, Hamilton and Pooch, 1995, Simons, 1998). The emphasis in this overview is on the modelling process, particularly for requirements specification, rather than the whole analysis cycle. The methods are categorised as first-, second- and third- generation after Simons (Simons, 1998) and Henderson-Sellers [OPEN, 2000 #124; (Henderson-Sellers and Simons, 2000).

2.6.1 First Generation Object-Oriented Methods

First generation methods are characterised by (Simons, 1998) as:

"... methodologies that appeared after 1988, naively using enriched entity-relationship models as the basis for implementation. Characteristically attribute-centred, with late process modelling."

Object Modelling Technique (OMT)

OMT (Rumbaugh et al., 1991) is a popular data-oriented method providing a simple, language independent approach. The method emphasises the data components and data modelling techniques. OMT involves three models similar to Graham's (1994) generic ternary model outlined above. These are described as follows (Blaha and Premerlani, 1998):

- Object model - an information/data model which describes the static structure of the system
- Dynamic model - a state model which describes the temporal interactions between objects in the system
- Functional model - a process model which defines the computations that objects perform.

The method has a rich, detailed notation which usually requires the use of CASE tools. There are three main phases: analysis, system design and object design (implementation design). Of interest from a requirements engineering perspective is the analysis phase as outlined by Hamilton and Pooch (1995):

- Build an object model including object model diagrams and a data dictionary
- Develop a dynamic model including state diagrams and global event flow diagrams
- Construct a functional model including data flow diagrams and constraints
- Verify, iterate, and refine the three models.

The Object Model notation is a variation on entity-relationship diagrams and includes generalisation, aggregation, and association. The Dynamic model notation is a form of state transition diagram based on Harel's statecharts (Harel, 1987), and the functional model notation is based on data flow diagrams. According to two of the original authors of OMT, Blaha and Premerlani (1998) in a publication that applies OMT specifically to database design, " *Our approach, OMT, stresses the importance of models and the uses of models to achieve abstraction.*"

Shlaer and Mellor Method

The Shlaer and Mellor (1988) method is one of the earliest approaches. It is heavily based on traditional data modelling and the normalisation of tables. Process modelling using Data Flow Diagrams is used to describe object behaviour and dynamic modelling is based on state transition diagrams and object life cycles (Henderson-Sellers, 1997). It has been criticised as not truly object-oriented (Graham. 1994, Winblad et al., 1990) because of the emphasis on information modelling at the expense of the *"... basic object-oriented philosophy of data-plus-functionality"* (Henderson-Sellers, 1997).

Coad and Yourdon's Object Oriented Analysis

Object Oriented Analysis (Coad and Yourdon, 1990, Coad and Yourdon, 1991) uses extensions to ER modelling and provides a five-layer model with the emphasis on analysis rather than design. There is progressive expansion of the analysis model towards implementation. There is no guidance on reuse or interface design and the method is often seen as an incomplete, data driven method omitting behaviour (Graham, 1994, Loosley et al., 1994).

The five stages or layers of object-oriented analysis are:

- Subject Layer - identify subjects, subsystems composed of 5 to 9 objects
- Class and Object Layer - describe objects in detail
- Structure Layer - identify inheritance structures
- Attribute Layer - specify attributes and relationships
- Service Layer - define services, specify methods.

Responsibility-Driven Design (RDD)

The Responsibility-Driven Design approach (Wirfs-Brock et al., 1990) is one of the earliest documented approaches. The RDD approach is based on classes, responsibilities, collaborations (CRC) cards (Beck and Cunningham, 1989). CRC cards are physical index cards which list a class, its responsibilities and its collaborators. These cards are used to support the abstraction process as candidate classes and objects are identified. This approach assumes existing user requirements and has the following phases:

- identify objects (from nouns in specification) and organise into classification structures
- find responsibilities (from verbs in specification)
- assign responsibilities to classes
- refine responsibilities based on classification structures
- find collaborations
- discard classes without collaborations
- refine structures
- group responsibilities into contracts
- use collaborations to define subsystems
- fill in the details.

2.6.2 Second Generation Methods

Second generation methods are characterised by Simons (1998) as:
" ... methodologies that appeared after 1991, synthetic approaches, borrowing techniques indiscriminately from each other and the first generation. Characteristically having rich notations, but little guiding process or coherence between stages."

Booch Method (1987/1991/1994)

Booch's (1987, 1991) early work was based on object-oriented programming languages, particularly Ada. The method is now oriented to C++ although it is language independent and supports detailed design issues and some real-time issues. The development process includes micro and macro lifecycles and is documented using scenarios, class diagrams, state transition diagrams and object diagrams. The method involves the following steps:

- identify classes and objects using separate diagrams
- identify semantics of objects and classes
- identify relationships
- implement objects and classes.

Later Booch (1994) improved the method and added interaction diagrams, a simpler notation, use-case analysis and the definition of a macro process as follows:

- establish core requirements
- develop model of behaviour (analysis)
- create architecture (design)
- evolve implementation
- post delivery maintenance.

Methodology for Object-Oriented Software Engineering of Systems (MOSES)

MOSES (Henderson-Sellers and Edwards, 1994, Henderson-Sellers, 1997) is a comprehensive method providing metrics, project management reuse and business modelling. It has five graphical deliverables and eight textual deliverables. It also provides a general, broad model for analysis and design with the following phases:

- system requirements specification
- identify candidate object/classes
- establish relationships
- analysis merges to design
- bottom up design using existing library classes - begin coding/testing
- introduce inheritance - finalise coding/testing
- clustering/generalisation.

In MOSES there is no distinct OOA/OOD boundary which is part of the philosophy of the "seamless" transition between analysis and design. Analysis is complemented by the synthesis of design while analysis continues. The models for analysis and design are the same. The system is described in terms of abstract data types (ADTs) and roles or Classes, Instances, Roles and Types (CIRTs). Systems are modelled as networks of methods and messages where the relationship between two CIRTs provides a connection channel for messages to be sent. Messages trigger methods that lead to changes in state of CIRTs.

Henderson-Sellers (1997) explicitly describes two aspects of modelling: static object modelling and dynamic modelling. Static object modelling is based on descriptions of CIRTs and descriptions of CIRT interactions. CIRTs are identified by looking for nouns in the specification as a first cut. Then objects are identified in terms of the services offerred (responsibilities). The notation

for CIRT model diagrams can be loosely described as ER-type. CIRTs have names, properties (attributes) and operations (methods). Optionally, business rules can be added to the diagrams. Relationships are descibed in terms of:

- generalisation (inheritance): IS-A
- aggregation (composition): IS-PART-OF
- association: USES-A

Dynamic Modelling in MOSES is based on objectcharts, transition specification tables, event diagrams and service contracts which describe the obligations and benefits of clients and servers.

MOSES is the forerunner of the current OPEN method discussed in third generation methods below.

Object-Oriented Software Engineering (OOSE)

The OOSE method is based on Objectory (Jacobson et al., 1992) and addresses the whole development life cycle. It is based on a supporting architecture; a step-by-step method to apply the architecture concepts; a process or scaling up of the method to industrial activity; and tools to support the architecture, method and process. The tools are said to be analogous to those of the building trade (Graham, 1994). It is a complete and mature approach which supports detailed analysis and design processes with complete process documentation.

There are three types of objects: interface, entity and control objects and progressive, iterative cycles of analysis, construction, testing. Each iteration refines one version of the system to a new version. The defining characteristic of OOSE is the concept of the "use case", a concept which has since been incorporated into other methods (Jacobson and Christerson, 1995). A use case is

a description (or script) of how users interact with a system for specific transactions. This means that there is an infinite number of possible use cases for any system but pivotal ones can be used to guide the modelling and design. The OOSE method has six models:

- requirements model - use cases + actors (external entities)
- domain object model - objects based on business entities
- analysis model - interface objects, entity objects, control objects
- design model - blocks which are implementations of one or more objects
- implementation model - annotated source code
- test model - test specifications derived from use cases.

2.6.3 Third Generation Methods

Third generation methods are characterised by Simons (1998) as:
"... methodologies appearing after 1995, focussing on the technical process and the management process. Characteristically more selective in choice of techniques, inclusion of metrics and cross-checks."

Semantic Object Modelling Approach (SOMA)

Although SOMA was first published in 1994, Simons classifies it as a third generation method. SOMA (Graham, 1994) is based on ER concepts and Coad and Yourdon's (Coad and Yourdon, 1991) approach with the addition of layers and rules (from artificial intelligence). SOMA has seven activities:

- identify layers (sets of objects with semantics)
- identify objects
- identify usage, classification, composition structures

- define data semantics and associations
- add attributes to objects
- add operations to objects
- add declarative semantics (rules).

The OPEN method

OPEN stands for Object-oriented Process, Environment and Notation (Henderson-Sellers and Simons, 2000, Graham et al., 1997). According to Henderson-Sellers (1997) this method is a merging of SOMA, MOSES, Ptech *"...together with strong influences from RDD, BON, OOram, [and] OBA"*. The OPEN Web Page describes the method as *"...the premier third generation, public domain, fully object-oriented methodology/process"* [OPEN, 2000 #124].

OPEN is a mature method that is designed to deliver a complete and disciplined process for object-oriented software engineering. The method supports iterative and incremental development, promotes the Software Engineering Process (SEP) and is said to embody 30 person-years of effort (Henderson-Sellers and Simons, 2000). SEP is described as a time sequenced set of activities which transform a set of user requirements into software. The object model sequence of development is characterised by a requirements capture process followed by a logical design process (Simons and Swatman, 1997) in which three models are developed: a task object model, a system object model and an implementation object model. The first two models are based on business knowledge and the second two models are based on systems knowledge. The main constructs in the method are:

- Process units which define a set of related activities
- Activities which are defined in terms of a series of tasks
- Tasks which are accomplished using well-known object-oriented techniques

- Techniques include scenario analysis, CRC carding and object/class modelling.

Unified Modelling Language (UML)

UML is one of the best known of the third generation development methods. It arose out of a collaboration of the authors of the OMT, Booch and OOSE methods. According to the Rational/UML web site

"The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems."

A major objective of the developers of UML has been to have it adopted as a standard by the Object Management Group (OMG). The Object Management Group is an industry organisation of over 400 members who are committed to establishing broad agreement between vendors on terminology and standards. Members include DEC, Hewlett Packard, Prime, Sun Systems, IBM, and Microsoft. UML 1.1 was officially adopted as the object modelling standard by the OMG in November 1998. Although UML 1.3 (approved in June 1999) provides some corrections to problems of UML 1.1 (Kobryn, 1999), it has still been criticised for logical inconsistencies, technical errors and lack of clarity in its specification (Kobryn, 1999, Simons and Graham, 1998, Simons and Graham, 1999).

UML incorporates and extends OMT models, use case models and Booch models so that it includes nine diagrams:

- The Class diagram - a structural diagram that shows a set of classes, interfaces, collaborations and their relationships

- The Object diagram - a structural diagram that shows a set of objects and their relationships
- The Use case diagram - a behavioural diagram that shows a set of use cases and actors and their relationships
- The Sequence diagram - a behavioural diagram that shows an interaction, emphasising the time ordering of messages
- The Collaboration diagram - a behavioural diagram that shows an interaction, emphasising the structural organisation of the objects that send and receive messages
- The Statechart diagram - a behavioural diagram that shows a state machine, emphasising the event-ordered behaviour of an object
- The Activity diagram - a behavioural diagram that shows a state machine, emphasising the flow from activity to activity
- The Component diagram - a structural diagram that shows a set of components and their relationships
- The Deployment diagram - a structural diagram that shows a set of nodes and their relationships.

The processes involved in UML modelling are:

- Identifying the requirements of the application and modelling business processes
- Mapping requirements to abstract business objects, identifying and applying design patterns, and creating usage scenarios
- Identifying and designing business objects or partitioning services across a three-tiered services model
- Mapping business objects to software components and designing how components will be distributed across a network.

The Discovery Method

The Discovery method was developed by Anthony Simons in 1998 (Simons, 1998). He categorises it as a third-generation development method for object-oriented systems which *" ... draws on insights from both the data modelling and behaviour-centred schools of thought"* (Simons, 2000). It is described as transformational in that analysis models evolve towards design models. It is also described as having a cognitive focus based on the power of abstraction, selective use of techniques and "discovery processes" based on reinforcement. There are four main phases in the method:

- Task Modelling - interviewing, narrative modelling, system task identification and task scripting
- Object Modelling - data modelling (ER models), control modelling, CRC carding
- System Modelling - collaboration diagrams, coupling analysis, system layering, reuse of frameworks and components
- Language Modelling - lifetime analysis, attribute/relationship structures, method specification.

The emphasis of Discovery is that techniques and notations are *" ... borrowed from existing methods, but they are carefully selected for **fitness-for-purpose**"* (Simons, 2000). Discovery is not based on any particular software lifecycle model but encourages the assembling of appropriate techniques and tools for assisting the development process for the developer.

2.6.4 Summary of the Characteristics of Object-Oriented Methods

Tables 2.1 and 2.2 provide a summary of the characteristics of the methods outlined in sections 2.5.1, 2.5.2 and 2.5.3. All the methods outlined have some characteristics in common. These characteristics are dealt with in more detail in Chapter 4 as part of the development of a generic representation of object-

oriented requirements engineering in the form of an initial conceptual process model.

Method	Objects/Class models	Relationship/Association models
Shlaer & Mellor	Information Structure Diagram; Class Diagram	Information Structure Diagram; Subsystem Relationship Model
Coad & Yourdon	Class and objects diagram - layers 1,4	Class and objects diagram - layers 2,4
Responsibility Driven Design	Class card	Class card with collaborations
OMT	Object model	Object model
Booch Method	Class diagram; object diagram	
OOSE	Requirements model; analysis model	Requirements model; Analysis model
MOSES	Object/class model	Object/class model
SOMA	Object model	Object model
OPEN	OML Metamodel	OML Metamodel
UML	Class diagram; object diagram	Object diagram
Discovery	Data model	Collaborations; system layering

Table 2.1 A summary of object-oriented methods and their static models

Method	states & transition models	event models
Shlaer & Mellor	State Model; object life cycle	State transition diagram;
Coad & Yourdon	Object-state diagram; service chart	Service chart
Responsibility Driven Design		
OMT	Dynamic model; State chart	Dynamic model
Booch Method	State transition diagram	Event diagram
OOSE	State transition graph	Interaction diagram
MOSES	Object charts	Event model
SOMA	State transition diagram	State transition diagram
OPEN	State model	State model
UML	Statechart diagram; activity diagram	Event hierarchies; signals
Discovery	State diagram; activity diagram	State diagram

Table 2.2 A summary of object-oriented methods and their dynamic models

2.7 Summary

This chapter has reviewed the literature relevant to the general context of object-oriented requirements engineering which is the focus for this research project. This project investigates the use of object-oriented models in requirements engineering practice in order to define, understand and describe the role of object-oriented models in the requirements engineering process and to empirically validate concepts contributing to that role. Section 2.2 examined the literature with respect to the technical or process-related issues of requirements engineering. This material is the basis of the conceptual process model developed in Chapter 4. Section 2.3 provided an understanding of problem solving and modelling for requirements engineering by drawing on classical definitions of cognition and reasoning. This material contributes to the formulation of research questions in Chapter 3 and development of interview scripts used in the case studies described in Chapter 5. Section 2.4 presented published research into the requirements engineering process. Section 2.5 provided a definition of object-oriented approaches to system development and section 2.6 provided a review of several major object-oriented methodologies relating directly to modelling activities for requirements engineering. This review included summaries of several well-known object-oriented methodologies and their static and dynamic modelling methods. This material contributes to the specifically object-oriented characteristics of the conceptual process model developed in Chapter 4 and the research questions developed in Chapter 3.

The discussion of requirements engineering models and frameworks and the object-oriented methods introduced here are extended and described in considerably more detail in Chapter 4 as the basis of the development of a conceptual process model.

Chapter 3

Research Design

3.1 Ove.view

This chapter presents the research design for this project. Firstly, there is a discussion of commonly accepted philosophies, paradigms and methods adopted in information systems research. Secondly, the chosen research method is described in the context of the objectives of this research, the formal research questions and the unit of analysis used in the research project. Lastly, a justification of the choice of research methods used in each component of the research project is presented.

3.2 Research in Information Systems

Information Systems (IS) is still regarded as a relatively new area of research. Peter Keen (1987) described the mission of IS research as a study of *"... the effective design, delivery, use and impact of information technologies in organisations and society"*. There has been debate over whether information systems can be defined as a discipline in its own right due to its

multidisciplinary nature in that it draws its theory and methods from a number of existing disciplines such as computer science, management science, sociology, philosophy and psychology (Keen, 1991, Mumford et al., 1985, Banville and Landry, 1989, Kaplan and Duchon, 1988).

There has also been debate over the nature of information systems research and the appropriate methods for conducting information systems research (Mumford et al., 1985, Walsham, 1995, Fitzgerald et al., 1985). As a consequence of this debate several research method taxonomies have been published (Hamilton and Ives, 1982, Benbasat, 1984, Galliers, 1985, Galliers, 1992, Keen, 1991, Fitzgerald and Howcroft, 1998, Myers, 1999, Fitzgerald et al., 1985) and many approaches for conducting information systems research have been advocated and discussed (Galliers and Land, 1987, Galliers, 1992, Mumford et al., 1985, Myers, 1999, Fitzgerald and Howcroft, 1998, Hirschheim, 1985).

Information systems research is generally regarded as social research which can encompass positivist, interpretivist or combined philosophies (Galliers and Land, 1987, Galliers, 1992, Fitzgerald and Howcroft, 1998). To assist researchers in understanding and selecting potential research methods several taxonomies have been published which categorise research methods in information systems (Galliers, 1992, Fitzgerald and Howcroft, 1998, Myers, 1999).

Galliers (1992) divides information systems research approaches into two categories, scientific and interpretivist. (see Table 3.1) and provides a summary of these approaches in terms of key features, strengths and weaknesses. From this Galliers (1992) provides a non-prescriptive taxonomy which *"... provides a framework upon which to base questions as to the likely utility of alternative approaches in a given context."*

Scientific	Interpretivist
Laboratory experiments	Subjective/argumentative
Field experiments	Reviews
Surveys	Action research
Case studies	Descriptive/interpretive
Theorem proof	
Forecasting	Futures research
Simulation	Role/game playing

Table 3.1 Table 8.2 from Galliers(1992), page 149

Fitzgerald and Howcroft (1998) present a taxonomy of "Soft Vs Hard Research Dichotomies" (see Table 3.2). This taxonomy is a response to the issue that fundamental research philosophies are often seen as dichotomous. The taxonomy provides a multiple paradigm approach where different methods with complementary strengths could be used as appropriate.

PARADIGM LEVEL	
Interpretivist	Positivist
ONTOLOGICAL LEVEL	
Relativist	Realist
EPISTEMOLOGICAL LEVEL	
Subjectivist	Objectivist
Emic/Insider/Subjective	Etic/Outsider/Objective
METHODOLOGICAL LEVEL	
Qualitative	Quantitative
Exploratory	Confirmatory
Induction	Deduction
Field	Laboratory
Idiographic	Nomothetic
AXIOLOGICAL LEVEL	
Relevance	Rigor

Table 3.2 Table 3 from Fitzgerald and Howcroft (1998), p 160

Myers' taxonomy (Myers, 1999) is for specifically qualitative methods and identifies perspectives, methods and modes of analysis (see Table 3.3)

Philosophical perspectives	Positivistist research Interpretive research Critical research
Qualitative research methods	Action research Case study research Ethnographic research Grounded theory
Modes of analysis	Hermeneutics Semiotics Narrative and metaphor

Table 3.3 Myers classification of qualitative research methods

Research may be further categorised in terms of theory building, theory testing and theory refinement (Neuman, 1994). Theory building research is based on collecting data with a view to formulating a theory using inductive reasoning. It is often exploratory or descriptive and frequently uses qualitative and interpretive approaches. Theory testing research is based on collecting data with a view to providing evidence about the truth of a formulated theory, research question or hypothesis using deductive reasoning. It can involve descriptive and/or explanatory approaches. Theory refinement involves using the results of theory testing to refine and improve previously formed theories.

3.2.1 Paradigms and Philosophies in Information Systems Research

From the point of view of defining a research environment a paradigm encompasses a particular ontology or view of the world and its components and a particular epistemology or theory of knowledge and knowledge acquisition (Hirschheim, 1985). Social research is often defined in terms of positivist or interpretivist paradigms. Positivism attempts to apply scientific methods to social sciences and interpretivism attempts to understand and interpret *" ... how people create and maintain their social worlds."* (Neuman, 1994). Myers (1999) describes the positivist paradigm as one where it is assumed that *" ... reality is objectively given and can be described by measurable*

properties which are independent of the observer (researcher) and his or her instruments" and further that Orlikowski and Baroudi (1991) classified IS research as positivist *"... if there was evidence of formal propositions, quantifiable measures of variables, hypothesis testing and the drawing of inferences about a phenomenon from the sample to a stated population"*. The interpretivist research paradigm is described by Myers (1999) as one where it is assumed that *" ... access to reality (given or socially constructed) is only through social constructions such as language, consciousness and shared meanings ... [and] ... interpretive research does not predefine dependent and independent variables, but focuses on the full complexity of human sense making as the situation emerges* (Kaplan and Maxwell, 1994)."

Walsham (1995) considers interpretive research as an empirical approach which focuses particularly on human interpretations and meanings. He also acknowledges the iterative nature of interpretive research which *"...results in an iterative process of data collection and analysis, with initial theories being expanded, revised, or abandoned altogether."*

3.2.2 Qualitative and Quantitative Research Methods

In simple terms quantitative methods are based on the collection and analysis of quantitative data (numbers) and qualitative methods are based on the collection and analysis of qualitative data (text, pictures, artifacts) (Neuman, 1994, Miles and Huberman, 1994).

An important concept (Neuman, 1994) is the notion of the empirical nature of collected data:

> *"Scientists gather data using specialised techniques and use the data to support or reject theories. Data are the empirical evidence or information that one gathers carefully according to rules or procedures ... Empirical evidence refers to observations that people*

> *experience through the senses - touch, sight, hearing, smell, taste. This confuses people, because researchers cannot use their senses to directly observe many aspects of the social world about which they seek answers (e.g., intelligence, attitudes, opinions, feelings, emotions, power, authority). Researchers have many specialised techniques to observe and indirectly measure such aspects of the social world." [Page 6](Neuman, 1994)*

This quote highlights the perceived problems (Galliers and Land, 1987, Galliers, 1992, Walsham, 1995, Hirschheim and Klein, 1992) of using purely quantitative approaches for research in social contexts. Quantitative methods are often based on traditional scientific method which is characterised by repeatability, reductionism and refutability (Galliers, 1992, Neuman, 1994). *Repeatability* assumes that an "experiment" can be replicated in exactly the same context. This is not true for social research if different individuals are involved. *Reductionism* assumes that a problem can be divided into manageable parts without distorting the issues. This is difficult where a researcher has evidence only about specific individuals and uses it to try and explain macro-level events (Neuman, 1994). *Refutability* assumes that there are verifiable predictions of anticipated outcomes and that the research process itself will not affect those outcomes. This is not often the case in social research where the researcher often explicitly influences the research (Walsham, 1995).

Understanding a phenomenon from the point of view of the participants in its particular social context is difficult to achieve when data is quantified (Kaplan and Maxwell, 1994). According to Myers "*The motivation for doing qualitative research, as opposed to quantitative research comes from the observation that, if there is one thing which distinguishes humans from the natural world, it is our ability to talk!*" Allowing participants to talk about and describe the situation under study provides a researcher with extremely rich data (Walsham, 1995).

Klein and Myers (1999) suggest that *"It is important to explicitly define what we mean by interpretive research. This is especially so given that no clear distinction is often made between 'qualitative' and 'interpretive' research... the word 'interpretive'"..." 'is not a synonym for 'qualitative' - qualitative research may or may not be interpretive, depending on the underlying philosophical assumptions of the researcher."* Qualitative methods deal with descriptive data, concepts, attitudes, beliefs etc and empirical data gained from experience (Neuman, 1994) and this data can be analysed in an interpretivist or a positivist manner. The interpretive paradigm is an overarching philosophy and deals with knowledge gained through social constructs - language, consciousness, shared meaning, documents and artefacts (Klein and Myers, 1999). Walsham (1995) suggests that there are two levels of interpretation in that as researchers we are *"...accessing others' interpretations through our own cognitive apparatus and feeding a version of events back to others"*.

3.2.3 Candidate Qualitative Research Methods

Once a philosophical perspective has been chosen for a research project a strategy of inquiry can be selected *"...which moves from the underlying philosophical assumptions to research design and data collection"*. (Myers, 1999). The strategy chosen has been called a "method" (Myers, 1999) or an "approach" (Galliers, 1992). The choice of research methods will be influenced by the underlying philosophy and the research objectives. The methods particularly appropriate for studying technology in its human context (as in this project) are conceptual study, action research, case study research, grounded theory and ethnography (Myers, 1999).

Choosing research methods for qualitative data and interpretive analysis is difficult. Miles and Huberman (1994) suggest that qualitative researchers are constantly adapting and modifying commonly used methods and that *"no study conforms exactly to a standard methodology; each one calls for the*

researcher to bend the methodology to the peculiarities of the setting" and that in fact the quest for qualitative researchers is the "*... creation, testing, and revision of simple, practical, and effective analysis methods*".

Conceptual Study Research

The conceptual study research approach (Keen, 1991, Shanks et al., 1993) also called the argumentative/subjective approach (Galliers, 1992) "*...involves the articulation of subjective beliefs about an area of invetigation.*" (Shanks et al., 1993) and is based "*... more on opinion/speculation than observation, thereby placing greater emphasis on the role/perspective of the researcher.*" (Galliers, 1992). It can be used to review existing bodies of knowledge as well as actual situations. Building a theoretical or conceptual model to represent a body of knowledge based on the researcher's interpretation of the literature can be considered a conceptual study and is one of the research methods used in this project.

Case Study Research

Case study research can take many forms, single or multiple cases and can be carried out within a positivist or an interpretivist philosophy (Yin, 1994, Benbasat et al., 1987, Lee, 1989, Eisenhardt, 1989, Darke et al., 1998, Cavaye, 1996). Case study research aims to examine a phenomenon in its natural context (Cavaye, 1996, Yin, 1994) and aims to contribute to knowledge by relating findings to generalisable theory (Cavaye, 1996). The case study research method is one of the most popular methods in information systems research and is well-suited to understanding the interactions between information technology-related innovations and organisational contexts (Darke et al., 1998) as in this research project.

Eisenhardt (1989) describes case stu[illegible]ch in a similar manner as *"...a research strategy which focuses on understanding the dynamics present within single settings ... [which] ... can involve either single or multiple cases, and numerous levels of analysis."* Eisenhardt (1989) also explicitly addresses the issue of "theoretical saturation" or how to determine the appropriate number of cases. She suggests that multiple case design requires the study of at least four, but no more than ten cases. Closure can often be determined by looking at how much new information is likely to emerge from studying further cases (Cavaye, 1996).

Action Research

The action research method involves collaboration between the researcher(s) and the participant(s) and intervention by the researcher(s) in the situation being studied (Susman and Evered, 1978, Susman, 1983, Baskerville and Wood-Harper, 1996, Baskerville and Wood-Harper, 1998, Avison et al., 1999). This often involves a cycle of feedback aimed at increasing the understanding of a given social situation (Hult and Lennung, 1980). Galliers (1992) describes action research as an interpretivist subset of the case study approach where the presence of the researcher affects the situation being studied. In their critical perspective of the method Baskerville and Wood-Harper (1996) describe it as *"... a paragon of the post-positivist research methods. It is empirical, yet interpretive. It is experimental, yet multivariate. It is observational, yet interventionist.... To an arch positivist it should seem very unscientific. To the post-positivist, it seems ideal."*

This research approach has many attractive characteristics for this project in terms of studying practising professionals and what they do and why. That is, active exploration, cyclic redefinition and reflective learning leading to a hermeneutic approach to analysis. Further, Baskerville and Wood-Harper (1998) suggest that there are many diverse forms of action research rather than

one definitive method. Subsequently a variation on the action research approach together with multiple case studies as described below in section 3.3.4 was the approach arrived at for this research project.

Grounded Theory

Grounded theory seeks to develop theory that is grounded in data that has been systematically gathered and analysed (Strauss and Corbin, 1990). A key element of the grounded theory approach is that the theory should be developed with a reflexive 'back and forth' interplay between data collection and analysis (Myers, 1999, Urquhart, 1998) without any preconceptions about the outcome until all the data has been collected and analysed. Glaser (1992) suggests that "*there is a need **not** to review any of the literature in the substantive area under study*" since it might contaminate the findings. There have been several successful applications of grounded theory in information systems published (Orlikowski, 1993, Pandit, 1996, Galal and McDonnell, 1998, Urquhart, 1998). Urquhart's research is heavily based on detailed conversational analysis which is particularly suited to the grounded theory approach.

Grounded theory was considered unsuitable for this project because of the project's exploratory nature in the investigation of a research domain in which there were preconceived notions from existing literature about the outcome. A grounded theory approach also raises the possibility of focusing on the "*micro phenomena of IS development*" that detailed grounded theory analysis might produce (Walsham, 1995).

Ethnography

Ethnographic research (also known as social anthropology (Miles and Huberman, 1994)) requires the researcher to spend a significant amount of time in the organisation or situation under study investigating day-to-day events,

behaviours and rituals. For this reason most ethnographic studies are of a single organisation or situation and are often long-term studies over months or years.

Since this research project focuses on the use of modelling techniques used by professional analysts, the setting for this research is a specific social interaction between an analyst and user(s) and subsequent reaction in terms of the models produced rather than the examination of a social situation or community over time in a particular work setting (Miles and Huberman, 1994). Therefore, this approach was considered inappropriate for this project.

3.3 The Research Framework

3.3.1 Research Objectives

The main objectives of this research are threefold and complementary. The first objective is to investigate the role of modelling in object-oriented requirements engineering practice. Secondly, this investigation contributes to a theory, based on a conceptual process model, about how practising professionals use object-oriented models and methods to specify information systems. Thirdly, the results of this study raise various questions for further investigation.

3.3.2 Formal Research Questions

The main objective of this research project is encompassed in the following research question:

"How are object-oriented modelling methods being used by practising professionals for requirements specification?"

The emphasis is on how practising professionals are currently using object-oriented modelling methods, not what the literature says they *should* be doing. In order to investigate this question it was necessary to go out into organisations and interview professional analysts who are undertaking or have undertaken object-oriented requirements engineering for commercial scale projects.

This broad research question can usefully be broken down into three subquestions each addressing the use of models in one of the three processes of the requirements engineering process as outlined below:

*Is **elicitation** influenced by the use of object-oriented modelling methods?*

*When, how and for whom is **object-oriented modelling** undertaken?*

*How is **validation** performed on object-oriented models?*

3.3.3 Unit of Analysis

In order to understand the use of object-oriented models by practising professionals the bounds of the research and data collection must be set by determining the unit of analysis - the entity about which data will be gathered and the context of that entity within each case study. Yin (1994) suggests that the unit of analysis defines the "case" in a case study. He suggests five possible units of analysis:

- individuals
- decisions
- programs
- implementation processes

- organisational change.

The choice of the unit of analysis in a study is related to the way the questions and propositions are defined. In this project the main unit of analysis is the individual experienced requirements engineer or analyst, within the context of four other elements derived from Yin's elements (see Figure 3.1).

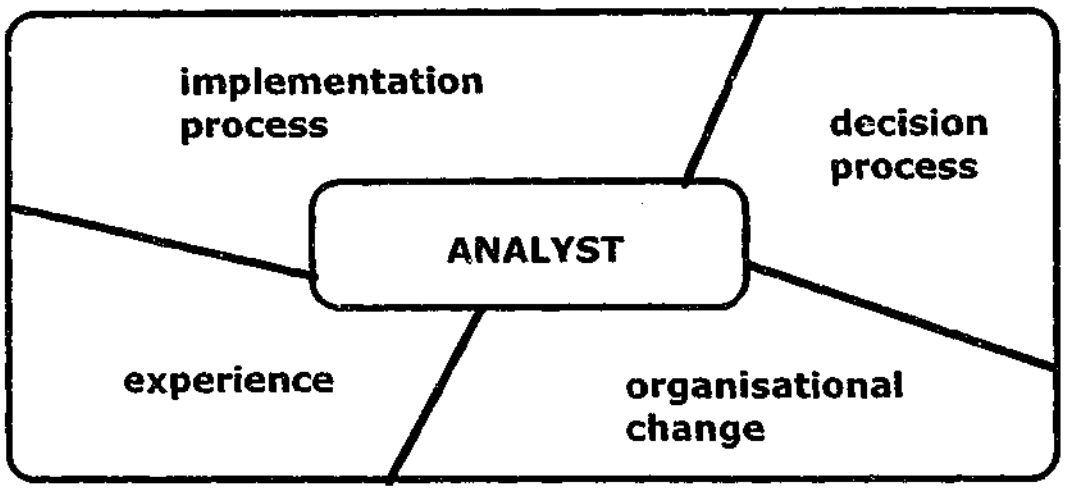


Figure 3.1 - Unit of Analysis - the analyst/requirements engineer in context

Implementation Process: the analyst is working in an organisation which has adopted object-oriented methods for system development. The analyst will be influenced by how the organisation has gone about implementing that change. The analyst may be one of many within the organisation who has been co-opted into the object-oriented program or he/she may be part of a pilot program within the organisation.

Decision Process: the analyst may have been consulted, or may not, about the decision to adopt object-oriented methods and may have been through an organisation sponsored training course.

Organisational Change: the adoption of object-oriented methods may have affected some aspects of the organisation's operation or may have affected the analyst's function or place within the organisation.

Experience: an individual analyst may have had specific training in object-oriented and non object-oriented requirements engineering methods and will have specific experience with one or more methods for commercial system specification and modelling.

3.3.4 The Research Approach

The research approach used for this project starts with a conceptual study and then continues with a method which is a hybrid of action research methods (Susman, 1983, Baskerville and Wood-Harper, 1996, Baskerville and Wood-Harper, 1998) and case studies (Cavaye, 1996, Eisenhardt, 1989, Yin, 1994). The hybrid approach is similar to case study research in that it relies on interviews and/or observations in the field, it is non-interventionary (and non-participatory), and it uses multiple sequential-cases. It is similar to action research in that it is iterative (each case has its own cycle), and it relies on learning and reflection on findings accumulated so far, both within individual case cycles and between cycles of multiple sequential-cases, for refining interview scripts and the conceptual process model. It differs from "action-case" (Vidgen and Braa, 1997) and action research in that there is no intervention or change as a result of the research process. The use of sequential-cases allows for cumulatively building a theory based on the rich structured picture that emerges, as more findings become evident. The intent is not to change what the participants do but to *actively* explore and discover what they do and why. The researcher is active in the data gathering process. The data is not gathered using neutral or passive techniques like protocol analysis (Ericsson and Simon, 1980, Sutcliffe and Maiden, 1992, Chaiyasut and Shanks, 1994) or "thinking aloud" protocols or recording interviews purely for later conversational

analysis as in grounded theory approaches (Strauss and Corbin, 1990). On the other hand, the actual protocols used by the participants in going about their work are of great interest, since it is these activities that the research approach is trying to identify and describe.

The method uses multiple sequential case studies which involve structured interviews with individual practising professional requirements engineers within the specific research domain. The research domain is set by the conceptual process model (presented in Chapter 4) which is grounded in the literature. Although a core set of interview questions remains the same for each case study (especially contextual questions), each case seeks to refine the conceptual process model by building on previous cases in two main ways:

- By testing for *reinforcement* of concepts already contained within the conceptual model
- By *revealing* new areas for exploration and potential reinforcement
- By *re-examining* previous interview transcripts to find any further reinforcement of an emerging category

Reflection on, and re-examination of data collected between cases leads to learning in the form of revelations and then to revised case documents and interview scripts. The cyclic and cumulative nature of the method allows for reinforcement of previous revelations. Reinforced concepts are retained in the conceptual process model. The process is ongoing but concluded when there has been enough reinforcement for a representative model of the research domain being investigated to stand alone or when theoretical saturation has been reached (Eisenhardt, 1989). Therefore, the outcome of the research method is a revised conceptual process model (with several revisions during the process) which represents a theory about the area being investigated which is initially grounded in the literature and then progressively grounded in data

gained from investigating the application of system development methods in practice.

The purpose of the method (Dawson, 1997, Carroll et al., 1998) is to provide an environment for the development of a theory grounded in the iterative and systematic gathering, structuring and classification of information. The starting point is a set of research questions exploring the opinions, beliefs and behaviours of the professional analysts and a conceptual process model representing the object-oriented requirements engineering process. The conceptual process model is iteratively refined in the light of accumulated findings and reflection on those findings. The accumulated findings from iterations within cases and from multiple cases contribute to a rich structured picture of a real world situation which contribute to a theory about the use of models in object-oriented rquirements engineering practice.

The Multiple Sequential-Case Research Cycle

The research approach is based on the theoretical foundations grounded in the literature as discussed in Chapter 2 of this thesis. The formulation of an initial conceptual process model (presented in Chapter 4) and research questions (described above in section 3.3.1) are based on the definitions established from the relevant literature. The conceptual process model together with the research questions is used to set up, plan and initiate the subsequent research cycle (Figure 3.2). This technique is adapted and extended from Miles and Hubermann (1994) who call this "focussing the collection of data".

Each case study research cycle takes place in a different real world or field situation. Each situation involves a different instance of the unit of analysis or practitioner (as defined above in section 3.3.3). Each research cycle involves the following activities: initiate the cycle, collect the data, evaluate the findings and go back to the field situation with a refined interview script and questions.

After the evaluation of findings at any iteration, the current accumulated findings and learning are reflected upon. This reflection activity provides two things: possible further refinements of the conceptual process model and possible refinements to the interview script in order to explore any emerging categories.

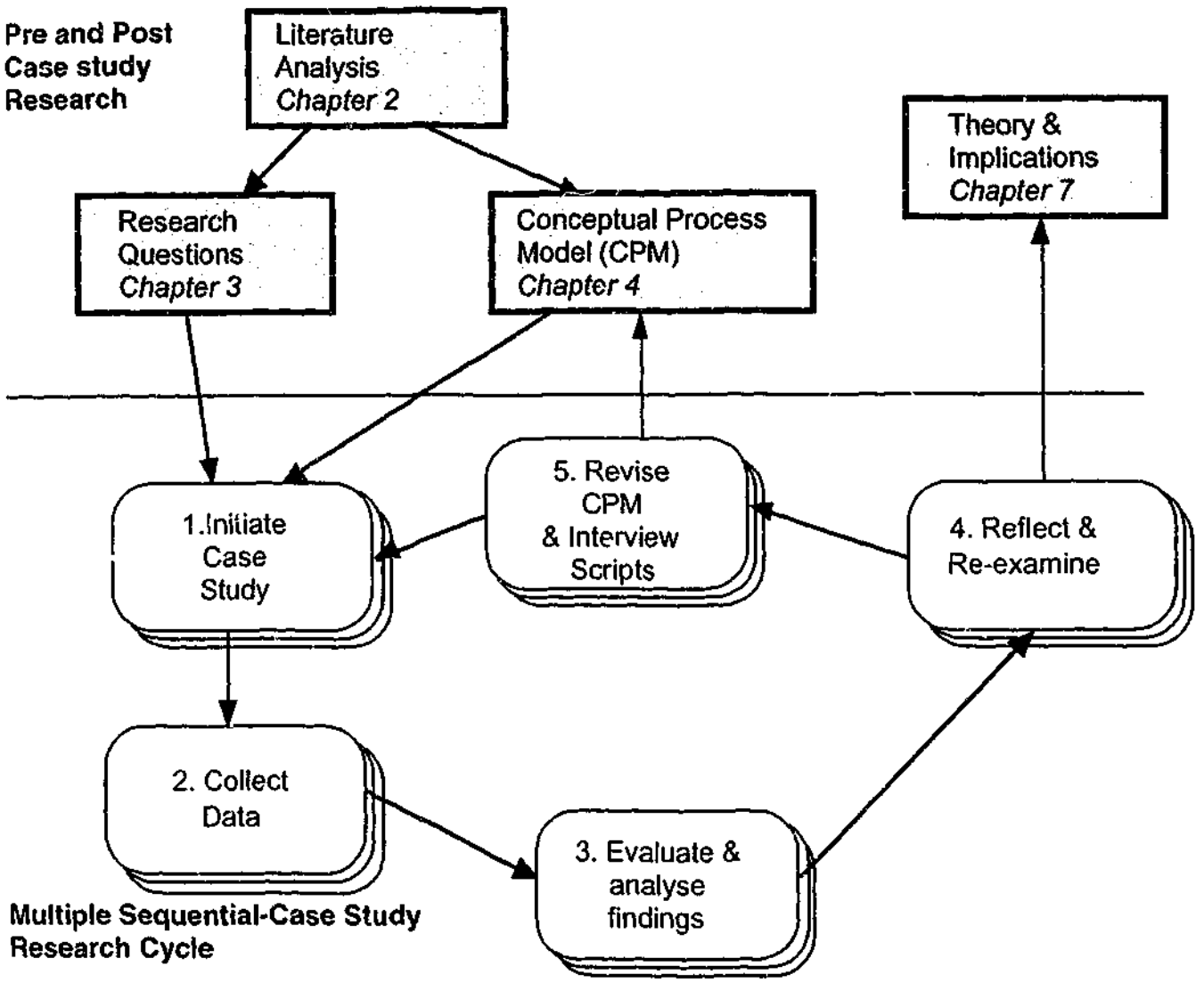


Figure 3.2 - Structure of the Research

As the number of cases increases and the accumulated data increases, an important part of the reflection process involves the re-examination of previous cases. This re-examination allows the potential reinforcement of emerging categories from the previous data.

Initiating the Cycle

The approach in this project was to use an evolving set of categories to structure the qualitative data as it was gathered. Researchers have suggested that, when using a qualitative approach, a set of initial *seed* categories may be generated to guide the research (Miles and Huberman, 1994). This approach has been used by Fitzgerald (1997) and Wynekoop and Russo (1997) in studies of systems development methodologies.

In this project a set of seed categories was formulated (detailed in Chapter 5, section 5.2.3) based on the research questions posed in section 3.3.1 above. These seed categories together with the initial conceptual process model led to the design of the initial interview script built around a set of categorised interview questions. Subsequently, the transcript of each interview was partitioned and inserted into a template structure reflecting the current set of categories. In each interview other categories and sub-categories emerged and were incorporated into interview scripts and template for investigation in the following cases. So the number of categories grew as the case dies continued.

Data Collection

Although the core of the data was gathered from taped and transcribed in-depth interviews, several other data sources were used. The case study protocol (described in detail in Chapter 5, section 5.2) provides five specific contact and/or clarification points between the researcher and the participant. An important aspect of this research method is that the case study protocol is designed for multiple sequential-cases. Reflection between and within case studies is critical to understanding, describing and categorising an accumulating body of data.

Evaluation and Analysis of Data

Analysis of the data is based on a template into which the raw transcribed data from the initial interviews was placed. The template was initially organised into seed categories based on the research questions described above. The transcription process brought to light emerging categories which were then incorporated into the template and subsequently into future interview scripts. The first step in the analysis process is the restructuring of the transcription into a template under headings representing the categories derived from the research questions and the conceptual process model. These headings provide an initial categorisation of the raw data without losing any of the richness, particularly useful quotes, from the original interview. Later the information from a follow-up interview is incorporated into the categorisation document where appropriate.

Reflection and Re-examination

Reflection in the form of examination of the data collected, its categorisation, and new categories or sub-categories emerging from the transcript data allows for a consideration of the kind of information being gathered and a consequential refining of the conceptual process model which in turn leads to modification of the interview script and template.

Revision of Conceptual Process model and Interview Script

The analysis and reflection on the data gathered for each case study leads to a modification of the interview script so that emerging categories can be explored in the next case study. Reflection also may result in the modification of the conceptual process model if evidence has emerged to warrant such a modification.

3.4 Justification of the Research Approach

The philosophical perspective taken in this research project is a broadly qualitative, interpretivist approach. The research domain centres on how practitioners use object-oriented models and methods to specify information systems. This project uses a descriptive model called a conceptual process model as a vehicle for developing a theory about the use of object-oriented models in requirements engineering practice. Data is inherently qualitative since it is gathered from professional requirements engineers by interviews. The analysis method is inherently interpretive since the findings are directly derived by the researcher interpreting the interview transcripts (Walsham, 1995). The position taken by the researcher is that of "outside observer" rather than "participant observer" (Walsham, 1995).

The approach to analysis taken in this project is broadly hermeneutic in that the analysis is attempting to discover the meaning of a text analogue as represented by interview transcripts. The use of sequential-case studies to revise a conceptual process model can be considered to be an instance of a hermeneutic circle where understanding is based on the movement from the whole to the part and back to the whole and where descriptions are guided by anticipated explanations (Gadamer, 1976). Although this research project uses interview transcripts as the raw data, analysis is not based on conversational analysis but on top down categorisations of how the transcripts represent a description of the situation under study. Therefore, the hermeneutic flavour of the approach is broad in the sense described by Klein and Myers (Klein and Myers, 1999) "*In Gadamer's description of the hermeneutic circle, the terms 'parts' and 'whole' should be given a broad and liberal interpretation. They can be parts of a historical story, and then the whole is the proper perspective of the historical context ...[the task] ...becomes one of seeking meaning in context.*"

The choice of research approach should be appropriate for the research being undertaken (Miles and Huberman, 1994, Neuman, 1994). As discussed in section 3.2 above several research approaches were considered. The chosen research approach described in section 3.4 above is a qualitative interpretive one comprising two components: a conceptual study and a set of multiple sequential-case studies using qualitative data. This section provides a justification for the selection of the chosen research methods within the research design.

3.4.1 The Conceptual Study

The conceptual study component of the research project involves the development of an initial conceptual process model of object-oriented requirements engineering grounded in the literature. As discussed in section 3.2. above a conceptual study can be used to review a body of knowledge and represent that body of knowledge in a conceptual model. The objective of the conceptual study in this research project is to develop a well-grounded conceptual process model representing object-oriented requirement engineering. The emphasis in this model is on how object-oriented models are used in object-oriented requirements engineering and how the models affect the processes involved. This conceptual process model provides a description of what the literature suggests is, or should be, happening in object-oriented requirements engineering practice and is used as a foundation for the case studies that follow.

The model is grounded in the literature and as such is one representation of the current state of theory about the object-oriented requirements engineering process, its processes, and the relationships between the processes and particularly the function of object-oriented models in the process and processes. This model is refined in the case study phase of this research project.

3.4.2 The Multiple Sequential-Case Study

The objective of the multiple sequential-case study component of this research project is to refine the conceptual process model and to address the research questions.

The refinement of the conceptual process model is based on the sequential-case studies which actively explore the components of the model with practising professional requirements engineers thereby revising and further developing the conceptual process model so that it represents object-oriented requirements engineering in practice. The emphasis is on the function of object-oriented models in the process and processes of requirements engineering.

The accumulated findings from iterations within cases and from multiple cases contribute to a rich structured picture of object-oriented requirements engineering as practiced which addresses the set of research questions exploring the opinions, beliefs and behaviours of professional requirements engineers. The main research question:

"How are object-oriented modelling methods used by practising professionals in the process of requirements engineering?"

is addressed by three subquestions each of which addresses the use of models in one of the three processes of the requirements engineering process:

*Is **elicitation** influenced by the use of object-oriented modelling methods?*

*When, how and for whom is **object-oriented modelling** undertaken?*

*How is **validation** performed on object-oriented models?*

Each of these research questions is addressed by questions in the interview scripts used in the case studies.

Qualitative data collection and analysis methods are necessary to investigate these questions. Ethnographic, action research and grounded theory approaches were considered inappropriate for the reasons outlined above. The nature of the research and the type of data needed to be collected determines the manner in which it is collected. In this case it was decided that the most appropriate method was sequential-case studies which could provide the accumulation of data necessary to revise and develop the conceptual process model so that it would provide a representation of requirements engineering practice. Case studies, particularly post hoc case studies such as those undertaken for this project, provide an appropriate way of presenting the researcher's interpretation of other people's interpretations as expressed in interviews (Walsham, 1995).

3.5 Summary

This chapter has presented and explained the research design used in this research project. Candidate data collection and analysis methods were discussed and the final choice of methods was justified in terms of the specific research being undertaken in this project.

The research was designed as theory building research and is based on two components: a conceptual study producing a conceptual process model grounded in the literature and a set of qualitative case studies which refine the initial conceptual process model based on findings grounded in practice and address research questions regarding the opinions, beliefs and behaviour of professional requirements engineers.

Chapter 4

An Initial Conceptual Process Model

4.1 Overview

This chapter develops an initial conceptual process model of object-oriented requirements engineering. Although there are several general requirements engineering process models and several object-oriented development methodologies described in the literature (and outlined in Chapter 2 of this thesis) there are no specifically object-oriented requirements engineering process models which exist independently of the system development methodology in which the requirements engineering process is being used.

The conceptual process model is proposed as a means of describing the process of object-oriented requirements engineering based on the literature in the two key areas: object-oriented models and methods; and general (non object-oriented) requirements engineering frameworks. The purpose of the initial conceptual process model in the research project is to provide a theoretical description of the object-oriented requirements engineering process as the basis

for subsequent fieldwork in the form of sequential-case studies. Interview scripts and data gathering methods will be explicitly based on the research questions and the view of object-oriented requirements engineering embodied in the model.

The conceptual process model builds on the definitions of key concepts within object-oriented requirements engineering as discussed in sections 2.5 of this thesis. The concepts in the model will be incorporated in interview scripts and then revised based on the findings from the fieldwork. This means that the model will go through several revisions in the course of subsequent fieldwork (described in Chapters 5 and 6 of this thesis) as it is modified to reflect the findings concerning object-oriented requirements engineering in practice. The final version of the conceptual process model will make a significant contribution to theory by providing a theoretical model for understanding and describing the specific domain of object-oriented requirements engineering as it is practiced.

This chapter provides a more detailed discussion of the concepts first introduced and outlined in Chapter 2 of this thesis. These concepts represent the two domains encompassed by a conceptual process model of object-oriented requirements engineering. That is, object-oriented modelling methods and requirements engineering process models and frameworks. Firstly, there is a deeper analysis of the object-oriented methods which were described in section 2.5 with a particular emphasis on modelling. The common characteristics of models and modelling activities used in twelve object-oriented methods are summarised in Tables 4.1a, 4.1b and 4.2. These tables provide considerably more detail than Tables 2.1 and 2.2. This detail provides the necessary foundation for the development of the conceptual process model. Secondly, several non object-oriented process models or frameworks for requirements engineering are described and analysed, again in greater detail than the outline provided in section 2.2 of Chapter 2. Finally, the concepts in these two areas are used to

develop an initial conceptual process model of object-oriented requirements engineering.

4.2 Categorising Object-Oriented Models for Requirements Engineering

All system development methodologies use various models in requirements specification. Object-oriented models have been categorised in the literature in various ways. Three major contributors to the categorisation of object-oriented models are Ian Graham (1994), Brian Henderson-Sellers (1997) and Anthony Simons (2000). The views of these authors are outlined in this section.

Regarding object-oriented methodologies, Graham (1994) suggests that the conventional wisdom in software engineering holds that a system should be described in three dimensions - data, process and dynamics or control. The data dimension corresponds to logical, static data models such as entity-relationship models. The process dimension covers Data Flow Diagrams (DFDs) or other process diagrams and the dynamics or control dimension is described by state diagrams or entity life history notation. Graham (1994) further suggests that there are three types of object-oriented methods, ternary methods, unary methods and hybrid methods of both types. Ternary methods mimic existing system development methods and have three separate notations for data, dynamics and process e.g. Object Modelling Technique, (Rumbaugh et al., 1991), Ptech, (Martin and Odell, 1992). Unary methods have one notation since the concept of an object inherently combines data and processes e.g. Responsibility Driven Design (Wirfs-Brock et al., 1990), Object Oriented Analysis (Coad and Yourdon, 1991). Hybrid methods encompass characteristics of both ternary and unary models e.g. MOSES, (Henderson-Sellers and Edwards, 1994), SOMA, (Graham, 1994).

Henderson-Sellers (Henderson-Sellers, 1997) classifies object modelling similarly but in two dimensions - static and dynamic. The static dimension includes the data and process modelling defined by Graham and the dynamic dimension corresponds directly to the dynamics and control dimension as defined by Graham. Henderson-Sellers (Henderson-Sellers, 1997) describes hybrid methodologies in terms of possible object-oriented/functional hybrid approaches. Hybrid methodologies can either be F-O-O approaches based on functional analysis which is transferred to an object-oriented view for design and implementation or O-O-F approaches where object-oriented analysis and design is followed by a functional (procedural) programming implementation (Henderson-Sellers, 1997).

Simons (Simons, 1998) proposes several models as part of the Discovery Method, a third generation development method for object-oriented systems. In the Discovery method design models evolve from analysis models and psychological reinforcement techniques are used to focus the developer's attention on relevant aspects of the system at each stage. Task modelling and narrative modelling, similar to use cases and scenarios, are heavily used. Data modelling in Discovery uses the rules of entity-relationship modelling, and responsibility analysis and interaction modelling is similar to the dynamic modelling described by Graham (Graham, 1994) and Henderson-Sellers (1997)

All of the models identified by these three authors can be categorised as either static data models or dynamic event and state transition models. These two categories are used in the following section to structure the discussion and summary of common modelling activities for object-oriented requirements engineering.

4.3 Static and Dynamic Modelling for Requirements Engineering

Several of the most commonly used object-oriented methods were described in detail in Chapter 2, section 2.5. An analysis of the characteristics of these methods reveals the following activities are often included in the **static object-oriented modelling process**:

- identifying and modelling objects and/or classes
- identifying and modelling object interaction and relationships between objects
- identifying and modelling inheritance relationships and hierarchies
- identifying and modelling responsibilities for objects and/or classes
- identifying and modelling processes and data flows
- modelling transactions using use cases, scenarios or task scripts.

Table 4.1a summarises the object/class, relationship/association and inheritance models found in static modelling approaches. Table 4.1b summarises the responsibility, process and use case/scenario models found in static modelling approaches. It should be noted that most of the methodologies categorise process models and use case/scenario models as static models as distinct from dynamic models which are generally seen as event or state-transition models.

Further analysis of the characteristics of these methods reveals the following activities are often included in the **dynamic object-oriented modelling process**:

- identifying and modelling object states and transitions
- event modelling
- modelling message passing and communication between objects.

Table 4.2 summarises the state and transition, event, and message-passing models found in dynamic modelling approaches.

Method	Objects/class models	Relationship or association models	Inheritance models
Shlaer & Mellor	Information Structure Diagram; Class Diagram	Information Structure Diagram; Subsystem Relationship Model	Inheritance Diagram
Coad & Yourdon	Class and objects diagram - layers 1,4	Class and objects diagram - layers 2,4	Whole-part hierarchy
Responsibility Driven Design	Class card	Class card with collaborations	Hierarchy graph
OMT	Object model	Object model	Object model
Booch Method	Class diagram; object diagram		
OOSE	Requirements model; analysis model	Requirements model; Analysis model	Inheritance hierarchy
MOSES	Object/class model	Object/class model	Inheritance model
SOMA	Object model	Object model	Object model
Discovery	Data model	Collaborations; system layering	Data model
OPEN	OML Metamodel	OML Metamodel	OML Metamodel
UML	Class diagram; object diagram	Object diagram	Object diagram

Table 4.1a A summary of object-oriented methods and their static models - part (a).

Method	Responsibility models	Process models	Use Case Scenario models
Shlaer & Mellor		Action data-flow diagram	
Coad & Yourdon		Class and objects diagram - layer 3	
Responsibility Driven Design	Class card with responsibilities		
OMT		Functional model	Scenarios
Booch Method		Process diagram	Life cycle script
OOSE	Analysis model	Graphical use case	Use case
MOSES	Service structure model		Scenarios
SOMA	Wrapper objects	Data flow diagrams	Task scripts
Discovery	Responsibility cards		Narrative model; Task scripts
OPEN			Scenario class diagram
UML	Object diagram; use case		Use case

Table 4.1b A summary of object-oriented methods and their static models - part (b).

Method	state & transition models	event models	Message passing models
Shlaer & Mellor	State Model; object life cycle	State transition diagram;	Object communication model; object access model
Coad & Yourdon	Object-state diagram; service chart	Service chart	Class and objects diagram - layer 5
Responsibility Driven Design			Class card with collaborations
OMT	Dynamic model; State chart	Dynamic model	
Booch Method	State transition diagram	Event diagram	Class diagram
OOSE	State transition graph	Interaction diagram	Interaction diagram
MOSES	Object charts	Event model	Event model
Ptech	Event diagram	Event diagram	Event diagram
SOMA	State transition diagram	State transition diagram	Layers and links
Discovery	State diagram; activity diagram	State diagram	Interaction diagram
OPEN	State model	State model	Collaboration diagram; Sequence diagram
UML	Statechart diagram; activity diagram	Event hierarchies;	Interaction diagram; message trace diagram

Table 4.2 A summary of object-oriented methods and their dynamic models

The static and dynamic models represented in Tables 4.1a, 4.1b and 4.2 are the types of models this research project expects to find being used in practice by professional requirements engineers. The concepts of static and dynamic modelling will be incorporated into the initial conceptual process model developed below in section 4.5.

4.4 Requirements Engineering Process Models and Frameworks

Having discussed the object-oriented modelling characteristics which need to be considered for the development of a conceptual process model of object-oriented requirements engineering it is also necessary to consider appropriate characteristics from requirements engineering process models and frameworks. As previously stated in Chapter 2 there are many definitions of requirements engineering (Loucopoulos and Karakostas, 1995, Macaulay, 1996) and a useful definition (Dawson et al., 1995) is:

> *"Requirements engineering is a process of elicitation, modelling, and validation of information system requirements which provides a specification which is the basis for the design and implementation of that information system."*

It is generally agreed that requirements engineering involves the analysis of a problem (Pohl, 1994) with a view to developing a set of unambiguous statements of requirements (Loucopoulos and Karakostas, 1995) describing what is to be designed rather than how it is to be designed (Macaulay, 1996). Traditional systems development approaches incorporate requirements specification as a separate phase within larger systems development methodologies. More recently there have been proposals for specific requirements engineering frameworks that can exist independently of the system development methodology with which they are used. The following sections describe some of the process models and frameworks used to describe the requirements engineering process (sometimes called systems analysis).

4.4.1 Traditional Systems Development Frameworks

Traditional approaches to requirements definition within systems development methodologies involve similar phases which attempt to understand a specific system development problem within a specific scope or problem domain and ultimately produce a specification or requirements document which is the basis of subsequent design and implementation (Avison and Fitzgerald, 1995). The traditional Systems Development Life Cycle (SDLC) developed and recommended to the National Computing Centre in the United Kingdom in the late 1960s (Avison and Fitzgerald, 1995) proposed three phases for requirements determination: feasibility study, system investigation and systems analysis. By the end of these three phases user requirements will have been elicited from clients, functional requirements will have been defined

and data and process models will have been developed for the design and implementation phases. Similar top down approaches often described as waterfall models such as Structured Systems Analysis (Gane and Sarson, 1978) and Structured Systems Analysis and Design Method (SSADM) (Downs et al., 1988) (see Figure 4.1), involve functional decomposition, feasibility or system studies and systems analysis which result in requirements definitions based on standard models such as data flow diagrams, entity-relationship models and structured English.

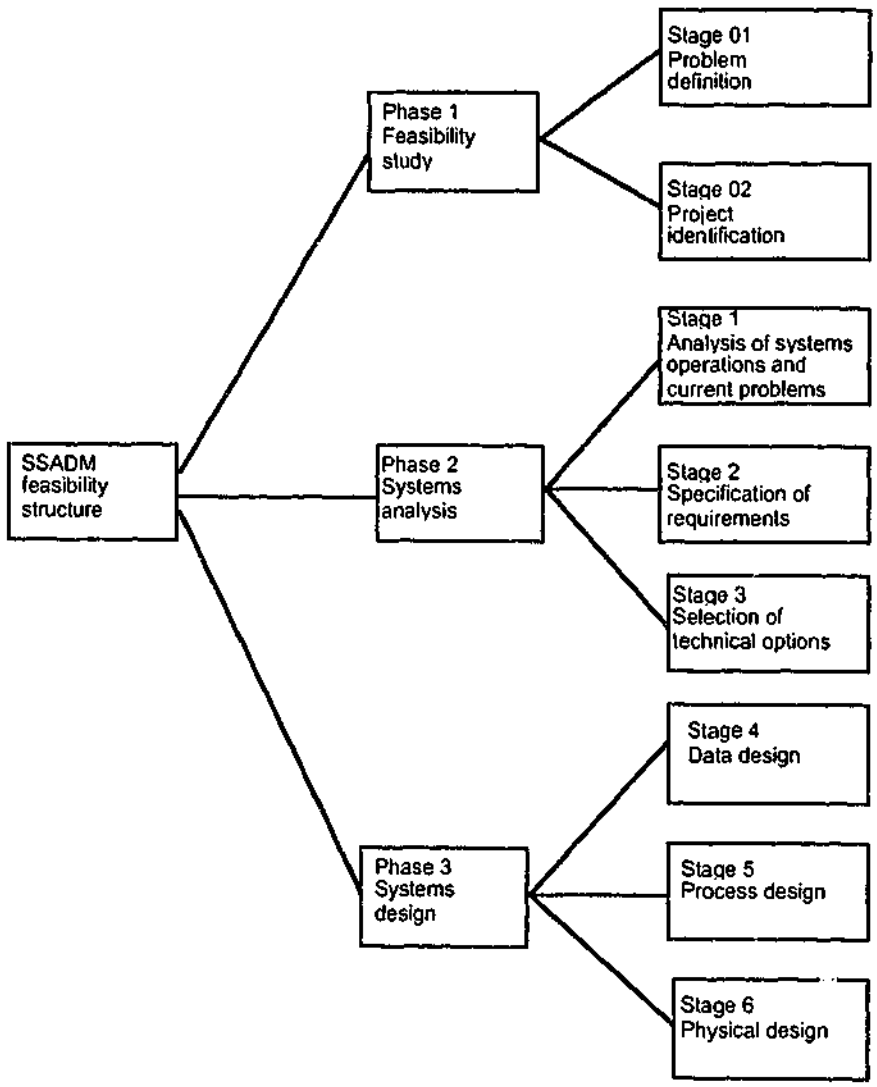


Figure 4.1 SSADM Feasibility Hierarchy (Downs et al., 1988) page 13

4.4.2 Viewpoint Approaches

Viewpoint approaches to requirements engineering emphasise the collaborative nature of requirements definition. These approaches specifically

address issues associated with conventional requirements capture techniques which *"...result in a linear communication structure in which each stakeholder group often understands and accepts responsibility only for the part of the application domain relevant to their specific interests."* (Darke and Shanks, 1997). Viewpoint development is the process of identifying and representing requirements from multiple stakeholder perspectives (Finkelstein et al., 1992, Darke and Shanks, 1996, Nuseibeh et al., 1994) and has been developed to support the requirements capture and representation processes by providing a mechanism for partitioning problem domains and accumulating domain information and requirements expressions. The Darke and Shanks model (Darke and Shanks, 1996) is reproduced in Figure 4.2. The elements of the model are:

- **Viewpoint development role**: identifies the intended use of viewpoint development - requirements acquisition, requirements modelling, or both phases
- **Viewpoint agent**: a particular role or view of the problem domain adopted by one or more stakeholders
- **Viewpoint representation**: an informal, semi-formal or formal representation of a viewpoint associated with a particular agent
- **Viewpoint development process**: defines essential activities carried out in viewpoint development - viewpoint identification, viewpoint representation, intra-viewpoint analysis and inter-viewpoint comparison.

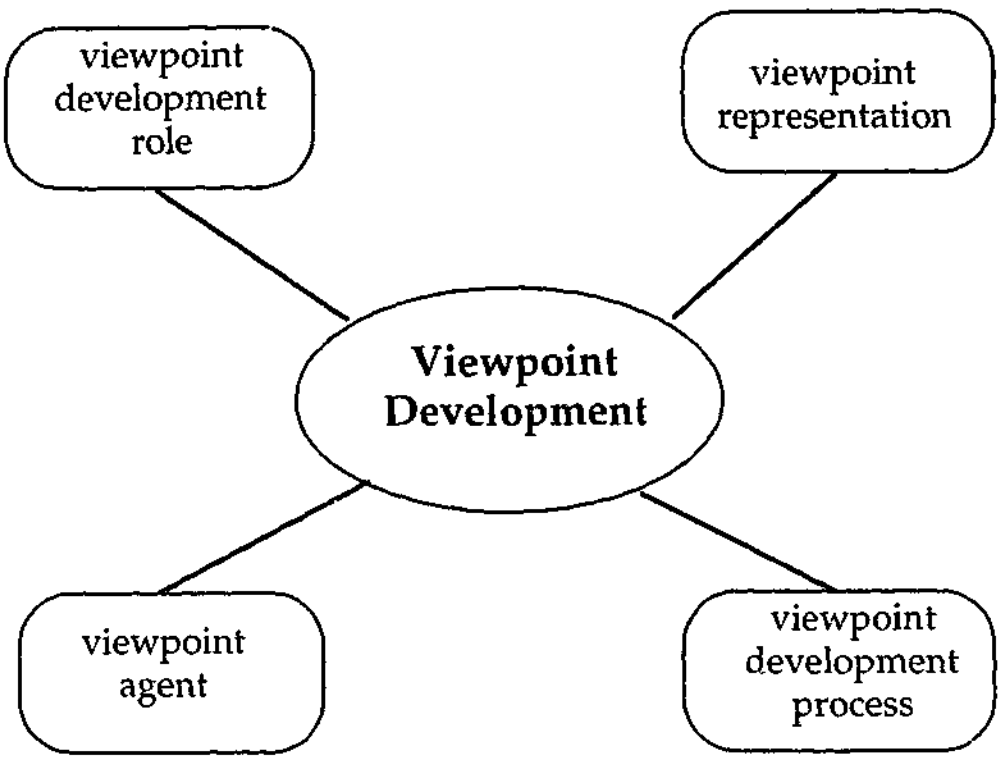


Figure 4.2 Viewpoint development framework (Darke and Shanks, 1996)

4.4.3 Macaulay's Model

Although Macaulay (1996) presents a general model of the requirements process (see Figure 4.3) she suggests that there is not one single process model and that *"...different situations require different process models"*. As an illustration Macaulay (1996) describes seven different models of the requirements engineering process in Chapter 6 of her book Requirements Engineering. These models are described in the context of a requirements engineer's "portfolio" of techniques. She suggests that the customer-supplier relationship will determine the nature of the requirements engineering process and that the requirements engineering process will in turn determine the contents of the portfolio. The different process models discussed include prototyping, soft systems methodology, Joint Application Development (JAD), Collaboration Responsibility Cards (CRC), focus groups and co-operative evaluation.

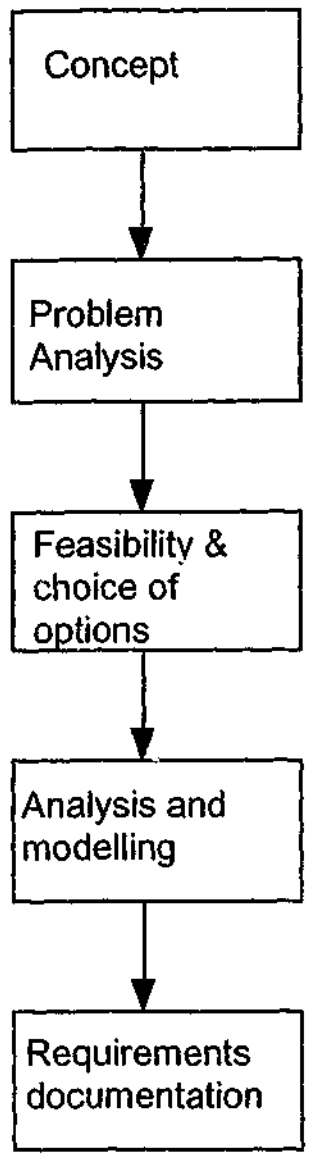


Figure 4.3 Macaulay's general process model, (Macaulay, 1996) page 7

4.4.4 Pohl's framework

Pohl (1994) regards requirements engineering as an interdisciplinary research area and presents a framework for requirements engineering which "*... can be applied to the analysis of existing RE practice ... a first step towards a common understanding of RE*". He proposes three dimensions of requirements engineering: specification; representation; and agreement, arising from a study of the literature.

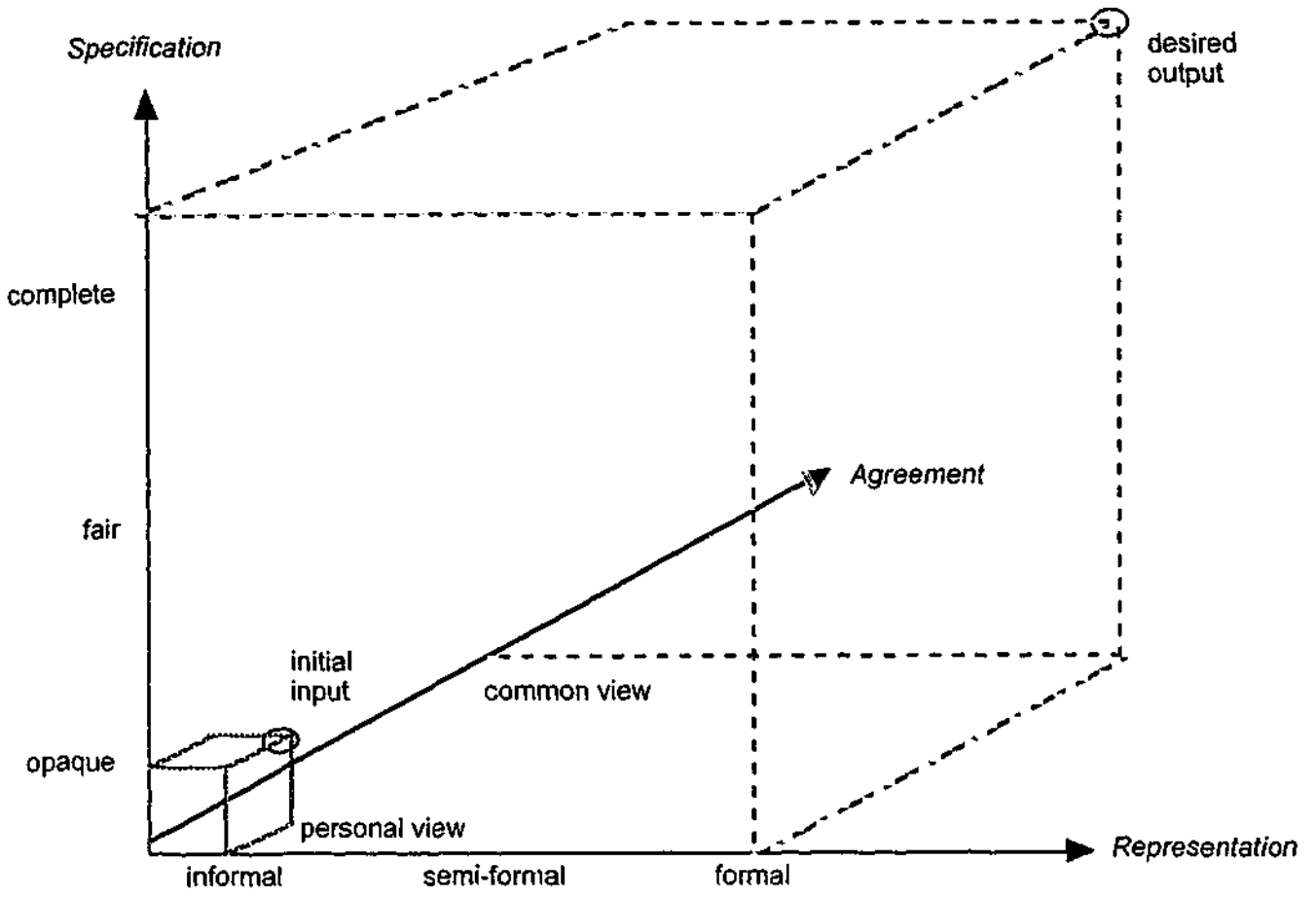


Figure 4.4 Pohl's three dimensions of requirements engineering (Pohl, 1994), page 246

Pohl's *specification dimension* (see Figure 4.4) involves taking a specification from the opaque (based on operational need) to a complete specification by an iterative process of definition and validation. The requirement specification should state *what*, not *how*, a system should be built and the specification should be unambiguous, complete, verifiable, consistent, modifiable, traceable and usable.

The *representation dimension* deals with possible different representations based on three categories:

- Informal languages - arbitrary graphics, natural language, descriptions by examples, sounds, and animations which provide the advantage of being user-oriented and expressive

- Semi-formal languages - ER diagrams, DFDs etc which provide the advantage of structural graphical visualisation, clear representation ("a picture is worth a thousand words"), a widely used quasi standard with some formal semantics which can be used for reasoning
- Formal specification languages – e.g. VDM, Z, ERAE, and Telos which provide the advantage of richer, system-oriented well-defined semantics from which it is possible to generate code.

Pohl's (1994) model suggests that a specification becomes more formally expressed as it is developed towards the final "desired output".

The *agreement dimension* has elements of the viewpoint approach discussed above in section 4.4.2 in that some requirements are shared, but many requirements exist only within the personal views of the people involved stemming from various roles (systems analyst, user, manager, developer etc). Pohl defines a *"common system specification"* or agreed specification as the ultimate goal of the process. That is, that the requirements engineering process tries to increase agreement. Co-existing specifications (from different personal views) are expressed using different specification languages. He further suggests that the identification of different views of the same system can have positive effects. That is, they can provide a good basis for requirements elicitation and assist in early validation and also in detecting additional requirements.

4.4.5 Loucopoulos and Karakostas' framework

A useful framework of the requirements engineering process has been proposed by Loucopoulos and Karakostas (1995). It is shown in Figure 4.5. In this framework the requirements engineering process can be broken down into three sub-processes; elicitation, specification and validation, which deal with two external entities; the user and the problem domain.

The purpose of elicitation is to obtain as much knowledge as possible about the problem in order to build a specification for the solution to the problem. Input comes from the user and existing information about the problem domain. Input to the requirements elicitation process includes:

- Specific user requirements
- Requirements of other stakeholders in the larger system (e.g. an organisation) which will host the software system
- Domain expert knowledge including literature about the domain
- Existing and similar software systems in that domain or other domains

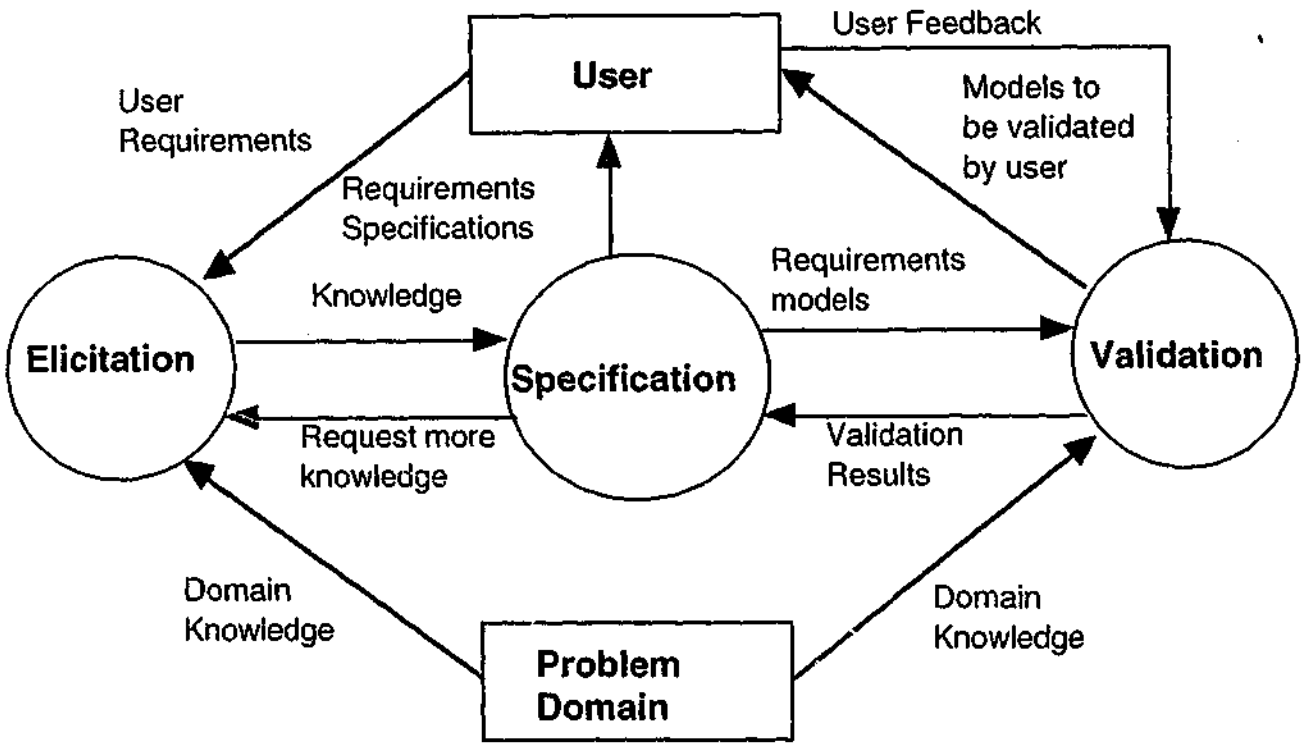


Figure 4.5: A framework for requirements engineering processes: Loucopoulos and Karakostas (1995) page 21

The deliverables from the elicitation process are usually not prescribed but Loukopoulos and Karakostas suggest that the whole requirements engineering process is a model creation process and the outcomes of the elicitation process are the conceptual models which are domain-dependent. As the requirements

engineering process progresses, the conceptual models are seen to become more software oriented than problem domain oriented.

The specification sub-process provides specifications and models for validation by the user and against the original problem domain. The purpose according to Loucopoulos and Karakostas is twofold:

- The specification model is used as an agreement between the software developers and the users on what constitutes the problem which must be solved by the software system
- The specification model is also a blueprint for the development of the software system

This process involves both analysis of requirements knowledge and synthesis of the knowledge into a coherent and logical requirements model.

The validation process is an ongoing process that proceeds in parallel with elicitation and specification and aims to ensure consistency, accuracy and relevance in the current models with the client's requirements. The validation process usually involves setting up and performing tests on the current version of the requirements model. The aim of validation is to produce a requirements model that is consistent with the users' requirements and expectations.

In all three sub-processes, one process does not end when the next process starts. Rather, each process relies on feedback from the other processes throughout the requirements lifecycle.

4.5 An Initial Conceptual Process Model

The fieldwork for this research project is based on the refinement of a conceptual process model of object-oriented requirements engineering. The initial conceptual process model proposed in this section contains characteristics of object-oriented methods as described in the literature and summarised in Tables 4.1a, 4.1b and 4.2 above within a framework based on the common characteristics of several requirements engineering process models described in section 4.4 above. All the models or frameworks discussed in section 4.4 include some or all of the activities, or processes, of knowledge elicitation (or knowledge acquisition), requirements modelling (or requirements representation) and requirements validation. These concepts are incorporated into the initial conceptual process model. The model is most heavily influenced by the framework proposed by Loucopoulos and Karakostas (1995) because their framework deals specifically with models and the modelling process which are concepts addressed in the research questions of this project.

In the traditional SDLC process model, requirements definition is just one phase of a larger methodology intended to address the entire system development process and as such does not provide the detail required for a requirements engineering process model. The viewpoint approach contains concepts of knowledge acquisition or elicitation and requirements representation or modelling together with a contextual element encompassing stakeholder or user requirements within some problem domain. This model was considered to be inappropriate for this research project and these research questions since the emphasis in viewpoint development is on the early stages of requirements definition and the framework is designed to support acquisition and/or modelling activities rather than provide a description of the modelling process itself within the requirements engineering process.

Macaulay's (1996) general model does not provide modelling of the interaction with stakeholders or users and the specific problem domain. This is necessary in understanding the use of models and the modelling process. Pohl's (1994) framework is comprehensive in terms of interaction and multiple dimensions for understanding the requirements engineering process but, like the viewpoint approach, it is not a true process model in that it does not provide a description of the modelling process itself within the requirements engineering process.

The framework proposed by Loucopoulos and Karakostas (1995) provides a comprehensive view of the requirements engineering process. It describes the requirements engineering process as a *"model creation process"* as discussed in section 4.3 above and incorporates a description of where the models are produced in the process. It also incorporates most of the characteristics of the other process models/frameworks, including the view of the users or stakeholders within a specific problem domain and an identification of the specific processes of elicitation, modelling and validation. The emphasis in this project on the modelling aspects of the requirements engineering process as embodied in the research questions points to the Loucopoulos and Karakostas (1995) framework as being the most suitable process model to use as the basis for the development of an object-oriented requirements engineering process model to be used in field research.

The initial conceptual process model proposed in this research project, revised from (Dawson and Swatman, 1998) and adapted from Loucopoulos and Karakostas (1995), is presented in Figure 4.6. This model represents the requirements engineering process as consisting of three processes: elicitation, modelling and validation, which interact with users and a specific problem domain. Each of these processes is described in detail later in this section. Information and artifacts (e.g. model diagrams, documentation etc) produced in

one process are often fed back into other processes for clarification or refinement.

The initial conceptual process model in figure 4.6 should be read in conjunction with the associated legend in figure 4.7. Much of the descriptive power is in the symbols used in the model and depicted in the legend.

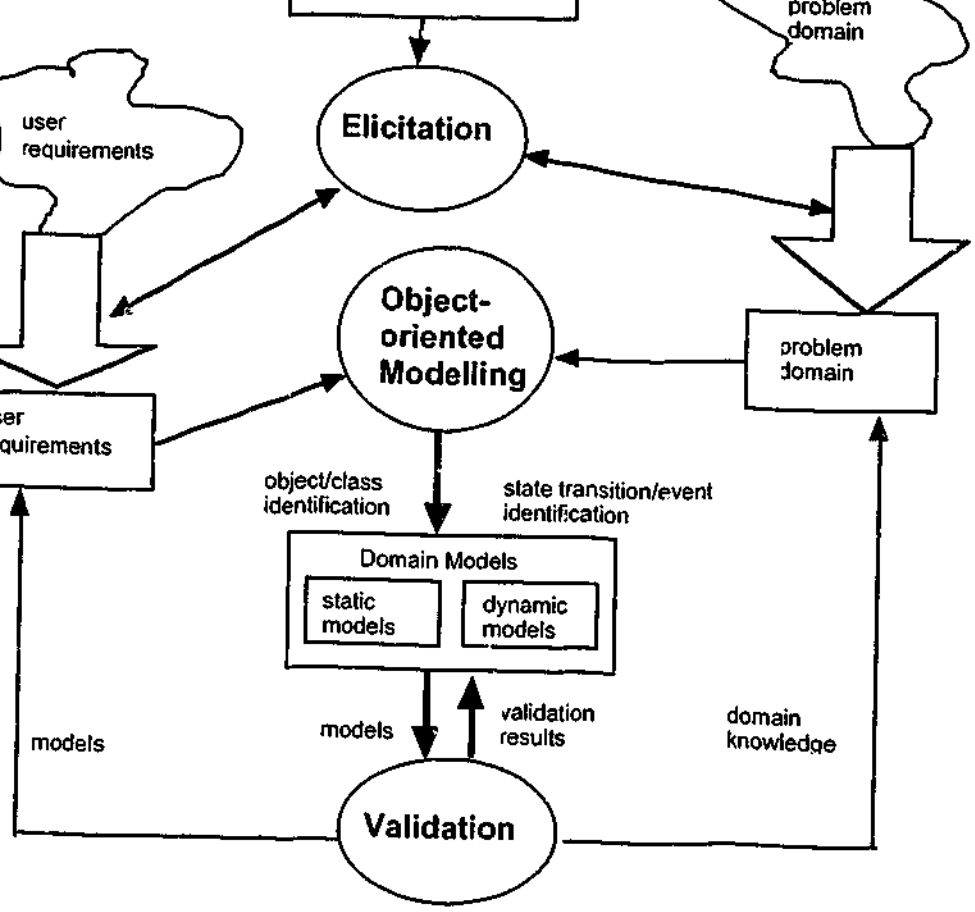


Figure 4.6 The Initial Conceptual Process Model

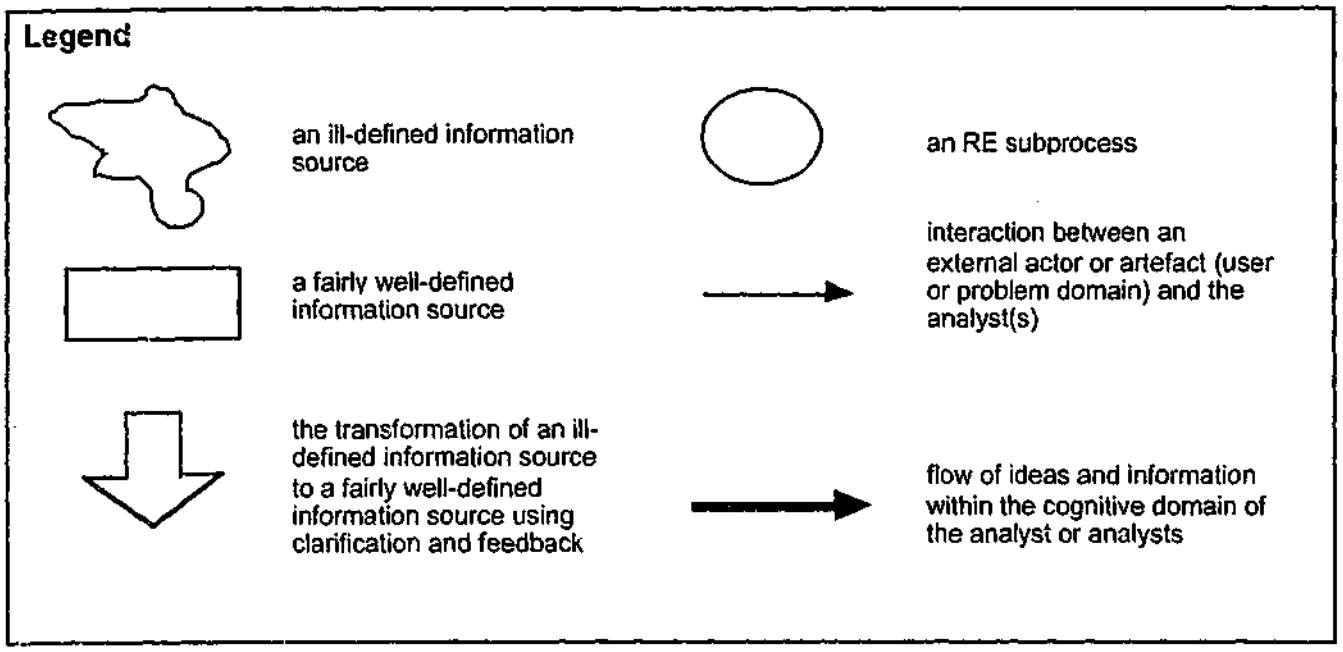


Figure 4.7 Legend for the Initial Conceptual Process Model

Each of the processes of the requirements engineering process is described in the remainder of this section.

4.5.1 The Elicitation Process

This process is based on input from three sources:

- knowledge from (as yet) ill-defined user requirements;
- knowledge from an ill-defined problem domain;
- predetermined and available resources such as hardware and software already available to the client or methods and tools already familiar to the client and the analyst.

These sources are implicit in the Loucopoulos and Karakostas framework as feedback loops and constraints based on existing systems, standards, and the interests of other stakeholders.

This elicitation process involves the transformation and clarification of user requirements and problem domain knowledge based on feedback from both domains until a view suitable for modelling is developed. This transformation is represented explicitly in the model by the hollow arrows. This transformation is analogous to the progressive resolution of an image being downloaded from the Internet from low resolution, containing few pixels, to higher and higher resolution.

4.5.2 The Object-Oriented Modelling Process

The acquired knowledge about the user requirements and the problem domain (represented as rectangles in the model) is then used as the basis of the object-oriented modelling process. The initial conceptual process model explicitly shows the static and dynamic models that are often produced during object-oriented modelling. The static models represent information about objects, classes, responsibilities and processes and include use case or scenario models as summarised in Tables 4.1a and 4.1b. The dynamic models represent state transitions, events and message passing characteristics as summarised in Table 4.2.

4.5.3 The Validation Process

In the validation process the models produced during the object-oriented modelling process are validated against the user's original requirements based on the knowledge about the problem domain. The process of refining the models by validation is based on feedback. Feedback from the user may necessitate reassessment and revision of the models for further validation. This is shown explicitly in the model. Also explicit in this process is internal validation by the analyst or analysis team against the original user requirements and problem domain information which was formulated during the elicitation process.

4.6 Summary

This chapter proposed an initial conceptual process model to be used in this research project. This model represents what the literature suggests is, or should be, happening in object-oriented requirements engineering. The initial conceptual process model is developed from the static and dynamic characteristics of object-oriented modelling and Loucopoulos and Karakostas' general requirements engineering framework.

According to the research design this model will be progressively refined using sequential-case studies of professional requirements engineers. The consequence is that the final version of the model represents a view of modelling activities in object-oriented requirements engineering that is not only grounded in the literature but also grounded in practice.

The model is used in Chapter 5 to develop interview scripts and analysis templates used in the case study field work. The progressive development of several versions of the model is used in Chapter 6 as a method for demonstrating and representing findings from the fieldwork.

Chapter 5

A Multiple Sequential-Case Study of Modelling for Object-Oriented Requirements Engineering

5.1 Overview

The case study phase of this research project is intended to empirically validate the concepts embodied in the conceptual process model developed in Chapter 4 of this thesis. Six case studies of object-oriented requirements engineering in practice were conducted in order to investigate the applicability of the conceptual process model and to revise the model where appropriate based on the investigation of professional practice in requirements engineering. The case studies were carried out in an explicitly sequential manner so that preliminary analysis of the data from each case would contribute to the progressive refinement of the conceptual process model by building on previous case study findings. This progressive refinement was based on the following chain of activities: first, looking for reinforcement of concepts already contained within the conceptual model; second, by following up any new or emerging concepts

revealed by a case; and third, by re-examining previous cases to find any further reinforcement of an emerging category. This progressive data collection and analysis is integral to the research methodology and is explicit in the case study protocol described in section 5.2 below.

Whereas the initial conceptual process model developed in Chapter 4 of this thesis provides a representation of object-oriented modelling for requirements engineering as described in the literature, the research questions explore the opinions, beliefs and behaviours of professional analysts engaged in object-oriented requirements engineering. The research questions addressed by the case study phase were formulated as one main research question and three sub questions in Chapters 1 and 3 as follows:

How are object-oriented modelling methods used by practising professionals in the process of requirements engineering?

- *Is **elicitation** influenced by the use of object-oriented modelling methods?*
- *When, how and for whom is object-oriented modelling undertaken?*
- *How is validation performed on object-oriented models?*

As described in Chapter 3, a multiple sequential-case study was used in order to accumulate and progressively analyse data so that the conceptual model could be revised based on findings in terms of reinforced or new concepts, as they became evident. The purpose of this approach is to specifically incorporate current findings and accumulated data into each case rather than collect the same type of data from all the cases and analyse it post hoc in isolation. The case studies were based on one-to-one interviews with practising requirements engineers who variously described themselves as systems analysts, consultants

or developers. A case protocol was developed and based on the research method which was presented in section 3.3.4 of Chapter 3.

The two main documents used for collection and preliminary analysis of case data were structured open interview scripts, which were modified for each case based on the findings so far, and an analysis template containing specific categories into which raw transcription data were placed. Data were also collected before and after the main interview as described in section 5.2.3 below. The sequential-case studies focus on the use of models by practising professionals in the requirements engineering phase of systems development. The sequential-case approach

- allowed the data collected from each case to inform subsequent cases in the form of modified interview scripts and categorisation templates
- facilitated a proactive investigation of object-oriented requirements engineering by an active exploration of emerging concepts from cases as they became evident
- strengthened the research findings by allowing for later explicit cross-case comparison.

Data collected was qualitative in the form of interviews, and qualitative data analysis methods (Neuman, 1994, Miles and Huberman, 1994, Myers, 1999) were employed for the analysis of the case study data. The main data collection part of the interview was structured around the research questions and the three processes represented in the conceptual process model proposed in Chapter 4. The unit of analysis under study in any one case was a particular requirements engineer or systems analyst within the context of a specific project for a specific client. Interview transcripts and follow-up interviews provided a rich picture of the perceptions of a participant's view of the requirements engineering process and the concepts represented in the conceptual model.

Preliminary cross-case analysis (Miles and Huberman, 1994, Cavaye, 1996) identified similarities and differences in perceptions and provided categories and concepts for further investigation in subsequent cases.

This chapter explains the case study protocol and the process that was followed, including the selection of cases, the data collection methods used, and the structure of the case study contexts in the form of case study descriptions. The six case study descriptions are presented in sections 5.3, 5.4, 5.5, 5.6, 5.7 and 5.8 of this chapter. The full cross-case analysis of the case study data which resulted in the evolution of the conceptual model to its final version appears in Chapter 6. Chapter 7 examines the implications of the findings embodied in the final conceptual process model and proposes a theory for understanding the practice of both to object-oriented requirements engineering in particular and requirements engineering in general.

5.2 The Case Study Protocol

5.2.1 The Case Study Process

The six case studies were conducted in four different organisations during late 1998 and early 1999. For each case study the unit of analysis as described in Chapter 3 was a professional consultant acting in the role of a systems analyst or requirements engineer for a client outside the organisation. That is, for each case the participant was representing two organisations: his or her own consulting organisation and a client organisation for which the systems development project was being undertaken. The four consulting organisations participating in this study were small to medium consulting companies and the six client organisations were from various industry sectors. The system development projects themselves differed in size, in the way in which they were conducted, and the nature of the systems being built.

5.2.2 The Participants

As is common with this type of qualitative research, the case studies were opportunistically selected in that the participants were used because they were available and willing to be part of the study. Participants were recruited through industry contacts. Some participants provided contacts for subsequent participants. The lack of available professionals working in the field of object-oriented requirements engineering in Melbourne where this study was undertaken means that there was no attempt to select participants based on specific background characteristics. The common factor is that all the participants were currently working in the field of object-oriented requirements specification and had recently completed an object-oriented requirements specification which they were willing to discuss. Contextual information such as job description, number of years spent doing requirements engineering, number of years doing object-oriented requirements engineering, whether the consultant had undergone formal training was gathered for each consultant. This information and the projects each participant discussed are summarised in Table 5.1.

Case	Job Title	Years in RE	Years in OORE	Client	Project
1	Operations manager	3.5 yrs	0 yrs	Federal Govt	Complex Technical
2	Principal Consultant	15 yrs	10 yrs	State Govt	Web based transactions
3	Senior Consultant	12 yrs	4 yrs	Telecommunications	Fault Mgt System
4	Director & partner	22 yrs	13 yrs	Software developer	Insurance
5	Consultant	14 yrs	14 yrs	Software developer	Life Insurance
6	Technical Manager	12 yrs	5 yrs	Software developer	Stockbroking

Table 5.1 Background Information for each consultant

5.2.3 The Data Collection and Analysis Method

The approach to data collection in this project was to use an evolving set of categories to structure the qualitative data as it was gathered. Firstly, a set of seed categories (Miles and Huberman, 1994, Fitzgerald, 1997, Wynekoop and Russo, 1997) was formulated based on the initial conceptual process model (see Table 5.2) and these were used to formulate the initial structured interview script (see sample questions in column 2, Table 5.2 and complete script in Appendix A). The seed categories are based directly on the research questions posed in Chapter 3. These seed categories together with the initial conceptual model led to the design of the initial interview script built around a set of categorised interview questions.

Seed categories	Example questions
Elicitation How is knowledge elicitation influenced by the use of object-oriented modelling methods?	Is knowledge elicitation explicitly undertaken? Is it seen as object-oriented? Is it seen as sequential or opportunistic?
Modelling When, how and for whom is object-oriented modelling undertaken?	When does modelling begin? Which (how many) models are produced? How are the models used? Who are they produced for? Which models, if any, are shown to the user?
Validation How is validation performed on object-oriented models?	When does the validation process begin? Which models are used in the validation process? When is the validation process considered to be complete?

Table 5.2 Seed Categories and example questions

Subsequently, the transcript of each interview was partitioned and distilled into a template structure or categorisation document reflecting the current set of categories. In each interview new categories and subcategories emerged and were incorporated into interview scripts and categorisation documents for

investigation in following cases. So the number of categories grew as the case studies continued.

The analysis of the findings for each case was based on the following chain of processes:

- **Revelation** of emergent new categories, or sub categories, during an interview
- **Reinforcement** of previously emergent or seed categories by explicit questioning, during an interview
- **Re-examination** of previous interview templates and transcripts to find any further reinforcement of an emerging category

For example, the question *"Do you see knowledge elicitation as object-oriented? That is, do you start thinking in terms of objects while you are doing knowledge elicitation?"* in Case 3 led to a detailed discussion of mental modelling by the analyst. Questions addressing this idea, or emergent category, of "mental modelling" were incorporated into the subsequent interview scripts for further investigation in subsequent cases. Further, previous case transcripts were re-examined to see if evidence of mental modelling was apparent.

Although the core of the data was gathered from taped and transcribed in-depth interviews, several other data sources were used. The types of data, sources of data, and collection methods are summarised in Table 5.3

Type of data	Source of data	Collection Method
Overall contextual data	Initial interview	Made by phone and followed up by email to set the context of the project and the participant's role in the project
Background data	Core Interview	First part of taped interview
Qualitative data	Core Interview	Major part of taped interview
Clarification	Follow-up interview	Further questions in need of clarification which emerged from the transcription process
Final clarification	Summary report	A report submitted to the participant with a final request for comment from participant

Table 5.3 Types and sources of data and their collection methods

Firstly, an initial contact was made with a person in an organisation where requirements engineering might be taking place or, more often, where requirements engineering had taken place. This initial contact often resulted in setting up confidentiality agreements between the researcher, participant and the participant's client.

This phase also involved an informal discussion with the participant or contact person within the organisation outlining the research process that the participant was required to be part of. This often included preliminary information about:

- the participant's professional history
- the participant's place in the organisation
- some quantitative information on the client organisation and business area.

Another outcome of this first discussion was an agreed plan of action in terms of an agreed schedule and set of procedures.

The next step was to arrange an interview with the participant. The interview was a structured, open style interview using a script based on the research questions and the current version of the conceptual process model and usually lasted for forty minutes to an hour. The final annotated interview script is

included as Appendix A. The interview was taped with the participant's permission. All participants agreed to taped interviews. As soon as possible after the interview the interview tape was transcribed. During the transcription process a list of questions and clarifications were noted for a follow-up interview. This follow-up interview was conducted either over the phone or by email depending on the participant. Most participants preferred email.

The transcript of the interview text was restructured into a template, or categorisation document, under headings derived from the research questions and interview script. The transcription process brought to light emerging categories which were then incorporated into the template and subsequently into future interview scripts. These headings provided categorisation of the raw data without losing any of the richness, particularly useful quotes, from the original interview. Later, the information from the follow-up interview was also incorporated into the categorisation document where appropriate.

After an individual categorisation document was finalised a summary was written in "anonymous style" and sent to the participant for any final input or comment. "Anonymous style" is a writing style which presents the material in publishable state with regard to confidentiality so the participant can see how the material will be presented when published. The concept of anonymous style arose from Case 1 where the participant wanted to see how material concerning a highly confidential government client might be published. This style of presenting the final summary was explained to, and welcomed by, the other participants and was used in all other cases.

Finally, there was reflection on the data collected and its categorisation, particularly in the light of previous case studies. This allowed for a consideration of the kind of information being gathered and a consequential refining of the interview script, categorisation document and its set of categories.

This case study protocol provided at least four specific contact and/or clarification points between the researcher and the participant which added structure to the research process. An important aspect of this research method is that the case study protocol is designed for multiple sequential-cases. Reflection between and within case studies is critical to understanding, describing and categorising an accumulating body of data.

5.2.4 The Structure of the Case Study Descriptions

The data collected in the six case studies have been compiled into six case descriptions which each consist of five sections. The six case study descriptions appear as sections 5.3, 5.4, 5.5, 5.6, 5.7, and 5.8 of this chapter.

The structure of each case study description is as follows:

1. The Consultant and Consulting Organisation
2. The Client and the Project
3. The Development Methodology
4. Project Documentation
5. The Requirements Engineering Process
    - Elicitation
    - Modelling
    - Validation

The case study descriptions are structured into these five sections in order to reflect the structure of the interview scripts and the categorisation document. The first section deals directly with each unit of analysis or consultant in the context of their professional history and current job description with their employing organisation. The second section describes the project and any specific client-based requirements. The third section describes the methodology used in the particular project and how its choice and/or characteristics were

influenced by the nature of the project and the consultant's experience and preference. The fourth section describes any available project documentation including in-house documents, company information, manuals or technical documentation available to the development team etc. The fifth section addresses the requirements engineering process in terms of the three processes addressed in the research questions and embodied in the conceptual process model with particular emphasis on the role of modelling in the overall process and between processes. The progressive in-depth cross-case analysis and the evolution of the conceptual process model presented in Chapter 6 are based mainly on the data collected and presented in this section, although data from other parts of the transcript relevant to the evolution of the conceptual model was also incorporated.

The following sections present the six case studies undertaken for this project. The research method used, as described in Chapter 3, is cyclic in nature, where the data, preliminary analysis and learning from each case informs the subsequent case studies. For this reason the case studies are presented in the chronological order in which they were undertaken. The data includes data obtained prior to the main interview as well as data from follow-up interviews.

The presentation of each case is based on illustrated narrative style, or an oral narrative told in the first person, as described by Miles and Huberman (1994) and Myers (1999) and as used in Fitzgerald (1997) and Urquhart (1998). This approach is described as (Miles and Huberman, 1994) *"...each part of the sequence is followed by a series of illustrative excerpts [quotes from the transcripts]"* which does not resort to explicit coding but looks for *" ... key words, themes, and sequences to find the most characteristic accounts."* Where transcript data is quoted directly the researcher's questions or interactions are shown as bold italic and the participant's as plain italic.

5.3 Case 1: A New Infrastructure for a Federal Government Department Software System

Case 1 involved a small confidential project which had to be completed quickly for a government department. The system was highly technical and involved complex calculations and predictions. The consulting organisation in this case used a commercial semi object-oriented template method (Robertson and Robertson, 1997, Robertson and Robertson, 2001) which involved producing a set of requirements cards on which the specification document was based. A major consideration for the consulting organisation was the importance placed on clients accepting that the requirements engineering process is vital for the eventual working product.

5.3.1 The Consultant and Consulting Organisation

The consulting organisation was a small software development and consulting organisation consisting of a managing director, research and development manager, operations manager, business development manager and several software engineers. This is a small organisation so no one has a narrow role. Software engineers do requirements engineering, programming, documentation and presentations. The research and development manager was investigating current system development methods (in particular full object-oriented methods and notations such as UML) for adoption by the organisation. The business development manager is a recent addition to the senior staff and his role is to be proactive in seeking out new business for the organisation.

The consultant had been with the organisation for three and a half years. She spent the first two years with the organisation as a software engineer. Her current position is operations manager, in charge of the day-to-day running of the business. For this project the consultant had been asked to temporarily take

on the role of team leader, software/requirements engineer in order to train a junior programmer in both requirements engineering and the new methodology being adopted by the organisation (see below).

5.3.2 The Client and the Project

The project was initiated by public tender for a government department. The project had tight timelines and deadlines (two weeks for the requirement specification) because of a need to commit to budget expenditure. The consulting organisation recommended that the requirements should be tendered for first and that no quote should be accepted without a full requirements specification. This had been the policy of the consulting organisation for some years and the client agreed to tender for the requirements specification first and then re-tender for the development as evidenced by the following exchange (researcher's questions in bold, participant's responses in plain italic):

"So how was this project initiated?

It was a public tender which we responded to. We told them that we did not have sufficient information to give them a realistic quote and therefore could not supply a quote. We recommended that they didn't accept any quote until after a requirements analysis phase.

So you are actually doing that with all clients now? You are making a point of saying that a requirements specification is very important?

It is actually the area that we specialise in, but we just said to them, we recommend that you do not accept any quote - go into the requirement analysis phase then retender. It may be with us or someone else of your choice, but you are taking a lot of risk by not doing a requirement analysis. It turned out they

agreed and we are in the requirement phase, which is the documentation just produced and it is going back to tender now."

The client had a set of software modules that had been developed in an ad hoc manner over a number of years to provide complex probability calculations. The original implementation and structure of the software was complex and difficult to use and difficult to train others to use. The project required the specification of a user interface (or "wrapper") to link the software modules together. *"They cannot afford the time and resources to train an entire division so they want an interface written which will link all the modules together, make them usable and provide a lot of on line documentation and simplify the actual use of it as a tool."*

The nature of the project meant that the hardware and operating system environment was fixed since the calculation modules were not being rewritten. The software ran on a HP 9000 under UNIX and this environment was seen as a constraint identified in the requirements engineering process. The whole system was to support about 20 users in two different states.

At the time of interview the requirements document had been produced and handed to the client. The project was going back to tender for the development phase. The consultant felt that the more tenders there were for the development of the system then the more successful the team could consider the requirements specification.

5.3.3 The Development Methodology

The methodology used for this project is called Volere (Robertson and Robertson, 1997, Robertson and Robertson, 2001) which is a requirements engineering-only methodology, not a full systems development method. The methodology was new to the organisation and the consultant. The research and

development manager had been doing research into developing an in-house methodology based on current best practice and had attended a training course on Volere. He was sufficiently impressed to recommend the methodology for the organisation. Discussions took place between senior members of the organisation and it was decided to trial this method on a relatively small project with a view to making it the standard methodology for requirements engineering in the organisation. The methodology was described by the consultant as structured, easy to use, self-documenting and *" ... made it easy not to miss information."*

The methodology is based on a template and the use of cards to describe requirements. The template is a booklet that provides guidelines for the tasks which need to be undertaken during the process of requirements specification. There is also provision for modelling using use-case models and entity-relationship models. Every requirement that is documented is based on a requirements card (see Figure 5.1). The cards are filled out in collaboration with the client/users during the requirements specification process. Most of the characteristics are self-explanatory. One special characteristic is the "fit criteria" which is a user-defined test which ensures a requirement is a single functional unit which can be tested. In the words of the consultant

"You need to identify how you are going to prove how this requirement is being delivered. If you can't put a statement in there, or a test, then you have proven to yourself that it is not a tangible requirement. It's too ambiguous, it's not detailed enough, it is open to interpretation because you cannot definitely state how you are going to test it, and if you can't state how you are going to test it how are you going to deliver it. So it allows for identification up front which is something that can often be missed in interviewing ... the typical one has to be 'user friendly' whatever that means. How do you measure user friendly?"

Requirement No: Unique Number	**Requirement Type:** Section number from the template	**Event/Use Case No:** Event/Use Case Number/s related to this requirement
Description: A one-sentence statement of the requirement **Purpose:** Why is this considered important? **Source:** Who raised this requirement? **Fit Criteria:** Unambiguous test of whether a solution meets the requirement		
Customer Satisfaction: Degree of user satisfaction if the requirement is successfully implemented. Scale 1-5 (ambivalent to very pleased) **Dependencies:** Other requirements that use the same information, or have a change effect.		**Customer Dissatisfaction:** Degree of user dissatisfaction if the requirement is not met. Scale 1-5 (hardly matters to very displeased) **Conflicts:** Other requirements that disagree with this one - identifies a potential need for negotiation.
Supporting Materials: Pointer to definitions, models and documents that illustrate this requirement **History:** date created, changed, deleted, passed its quality checks, and a rationale explaining changes.		

Figure 5.1 –Volere Requirements Card (Robertson and Robertson, 1997)

The Customer Satisfaction characteristic is based on how *happy* the customer would be if the requirement *was* included and the Customer Dissatisfaction characteristic is based on how *unhappy* the customer would be if the requirement *was not* included. This allows the consultant and client to prioritise any "wish list" the client might have. For this project, with a short time frame, any requirement that could not be identified to be a core functional requirement was not included in the specification.

The cards are self-documenting and go straight into the requirements specification document. *"When the cards are complete a lot of the hard work is done."*

When asked to comment on the effects of the methodology on the organisation the consultant nominated the following points:

- it [ideally] removes some of the specialised skill from the RE process
- it provides structure and guidance via the template so that it *"removes the need for all knowledge to be in the head of one individual."*
- specifications should be more consistent
- the RE process should be more repeatable
- each specification should be as complete as possible
- it assists in training since people are being trained in a structured and repeatable process.

A technique used in the methodology is called "apprenticing" which involves the consultant (or one of the team - in this case the programmer being trained in requirements engineering) going into the client organisation and learning the existing system, whether it be paper-based or computerised. This technique was very effective in this project. The comment was that *"The issues of what we are trying to achieve become clear. Because to sit and look at a system you make assumptions but if you sit and use it, the purpose, the intention comes out."*

5.3.4 Project Documentation

The only documentation that the consultant was prepared to show the researcher was a blank template and a blank requirements card (see Figure 5.1) *"I cannot give you the actual one just done, but I can give you a blank template, that shows you the structure of what we are currently doing."*

5.3.5 The Requirements Engineering Process

Elicitation

The methodology starts with a "blast off" meeting where everyone (client/users and requirements engineering team) are brought together to describe the project in a high level manner. The idea is to get general information about the purpose and goals of the system - an overview rather than specific requirements. Among the issues addressed (using the Volere template) are:

- purpose - *"if you can't identify why you are doing the project then you shouldn't be doing it"*
- who is the client? - i.e. who is responsible for the system and who is paying for it.
- what is a successful solution worth to the client - *"if no tangible benefit can be identified then why do it?"*
- what happens if we don't solve the problem - *"if we don't solve the problem then why do it?"*
- who are the users - users are categorised into levels of expertise and degree of importance
- naming conventions for data elements - in this case quite specific technical terminology
- relevant facts - the consultant described this as a *"dead section"*. The only relevant fact identified was that this was phase 1 of the project and that the possibility of phase 2 etc needed to be kept in mind during the requirements engineering process
- constraints - hardware and operational
- time frame - how long will the project take?

The consultant described the blast off meeting as providing an overview to the client - i.e. *"everything a client should know before going into any project."* The consultant also commented that some of the client participants in the blast off meeting seemed to feel that they were participating *"... in order to get the document done"* rather than work towards a solution. The next step after the blast off meeting was the apprenticing activity. The consultant commented that this was very successful in this project and that *"...watching was not enough - actual doing what they do is the only way to understand"*.

From the blast off meeting and the apprenticing the team was able to write up the scope of the project. *"From that we were able to write up the scope of the existing system and its environment and from there we went into actually identifying individual requirements. From that we actually got them to do screen dumps from the existing stuff and got them to just talk to us about what they want."* After the scope was documented the identification of specific individual requirements began, using the requirements cards.

The requirements engineering team held regular project meetings in-house where the research and development manager was used *" ... as a kind of a quality check, to see that we hadn't missed anything, ... because he had done the training course, he was in a position to say where he felt we had missed information and needed to ask questions"* and on site meetings every second day at the Department, where *" ... we sat and talked and documented what was learnt."*

Modelling

Although use-case diagrams and entity-relationship models were available and recommended within the method the analyst (possibly because she was new to object-oriented development) only produced models which were based on the completed requirements cards and a simple flow chart. All requirement

specification was done with the clients using the cards. *"We can't write requirements. They (the clients) have to tell us their requirements. We don't work with the cards, they do. ... There is no point in me trying to tell them what they need - they need to tell us."* This was the only area where the consultant felt that the methodology was ineffective. She felt that this was *"mainly due to me"*. She was not used to using the cards and had not had time to go through all the training material. She felt that the cards were *"...more a hindrance than a help"*. She found it easier to supplement the use of requirements cards by working with large sheets of paper (A3) on which the team could write requirements as they came up and draw diagrams. *"The cards were getting in the way. They (the clients) were too interested in what the cards were about and why we were filling them in."*

This project (possibly because it was a user interface to existing software modules) had no high level modelling diagrams. Although the Volere methodology uses standard models such as use-case diagrams and entity-relationship models none of these were used in this project at the requirements specification stage. The only diagram used by the team was a "flow chart" to describe the interface.

"... all we were trying to do was identify the way the interface goes together in system modules.

So the diagrams were really interaction diagrams?

Probably more like flow charts - flow of information.

Flow charts not diagrams or nothing like ...

A feel for flow of information through the system. Then first thing you need to do is ABC and then you need access to the next step, next step etc. It was just creating an understanding of what the process is.

The Logic. What about structure charts?

No, nothing of anything of a defined type of document.

The main product of the methodology was the requirements specification document based on the cards and the template.

Validation

The process of validation of the requirements specification started early with the definition of the fit criteria as described above. This definition for each requirement *"...validates that it is a workable requirement."* A project meeting was held in-house where there was a walk through by the research and development manager, the trainee and the consultant prior to presenting it to the client *" ... to identify anything that may be missed"*. Further validation was performed using a standard walkthrough approach by going through the document *"page by page ... every requirement"* with the client. Typical questions asked at this stage were:

"Is that what you meant?"
"Is that a fair statement of the requirement?"
"Is that what you intended to do?"
"Does the fit criteria describe what you are actually trying to achieve?"

"We write the cards with the client, then we go away and write the document. The cards are essentially self-documenting, but then we do a second level of checking [to ensure] that we haven't misinterpreted [anything] by actually stepping through it again. The only testing is the identification of the fit criteria."

So, according to the consultant the validation of the specification is based on the client's agreement to the fit criteria.

When asked how she felt about the process after the requirements document had been finally submitted to the client (for re-tender of the development

phase) the consultant commented that she enjoyed the process, found the Volere methodology well structured and *"the benefits would even grow with a bigger project"* and *" ... as always the requirements [document] when I delivered it, even though we were working to a tight time frame, I could have quite cheerfully spent another two weeks making sure we had plugged the holes, but we just did not have the time. That is always the case with requirements engineering, there is always something else you want clarifying and where do you draw the line?"*

5.4 Case 2: Developing a Transaction Specification Methodology for Electronic Service Delivery for a State Government Authority

Case 2 used object-oriented methods to develop in-house "lazy dog" templates of identified "common transactions" for multiple clients who could then do their own requirements engineering with the assistance of an IT liaison person. These templates were called "lazy dog" because they were partly completed specification templates containing use cases (both scripts and graphs) and OMT diagrams which the client (via the IT liaison person) could alter by addition of or striking out of appropriate elements.

5.4.1 The Consultant and Consulting Organisation

The consulting organisation is a business consultancy which provides IT consultancy and educational services to a broad range of clients, both from the public and private sectors. The organisation's philosophy is outlined on their web site as: "*We do not subscribe to a single, rigid methodology. Each assignment is treated as a unique challenge. We tailor our approach to meet the specific requirements of each client, drawing on a wide range of well-researched techniques and the combined experience of our consultants.*"

The organisation employs 50 staff in Melbourne, Sydney and Canberra, covering a wide range of business and technology disciplines. The structure of the organisation is built around senior consultants - experts in their fields with a minimum of ten years experience. The majority of consultants have postgraduate qualifications in business and technology disciplines and are regular contributors to conferences, publications and industry forums.

The consulting analyst interviewed for this study holds the position of principal consultant in the consulting organisation. She leads the e-business and online government practice and has been with the organisation five years. She has been involved in systems development for 15 years, both in the US and Australia and over that time has undertaken requirements specification for various projects including projects where she has been part of the complete system development process. She holds B.Sc. and M.Sc. degrees in engineering management.

The consultant has been developing object-oriented systems for about 10 years and is self-taught in object-oriented system design and has given courses on object-oriented system development. She feels that "*OO is a very natural way for me.*"

5.4.2 The Client and the Project

The client was a central support group within government which was helping or facilitating other departments or client organisations (similar to business units) with the implementation of on-line service delivery to the general public. The client organisations were independent organisations within the government that provide or sell products or services to the general public. Some services available to the general public included:

- Payment of municipal rates

- Notification of change of address
- Acquisition of a driver's licence
- Acquisition of a liquor licence
- Booking roadworthy vehicle testing
- Information about transport services
- Feedback to government agencies.

This created a three tiered client/user structure as shown in Figure 5.2. The client support group had its own clients who were the departments or client organisations and the "end clients" of the final systems were the general public who buy or use the products and/or services of the client organisations.

The client support group functions included:

- IT support for client organisations
- Information Management
- development of IT strategies for the whole of government
- development of IT policies and standards for whole of government
- development of system development toolkits for client organisations
- purchasing of group licences for whole of government.

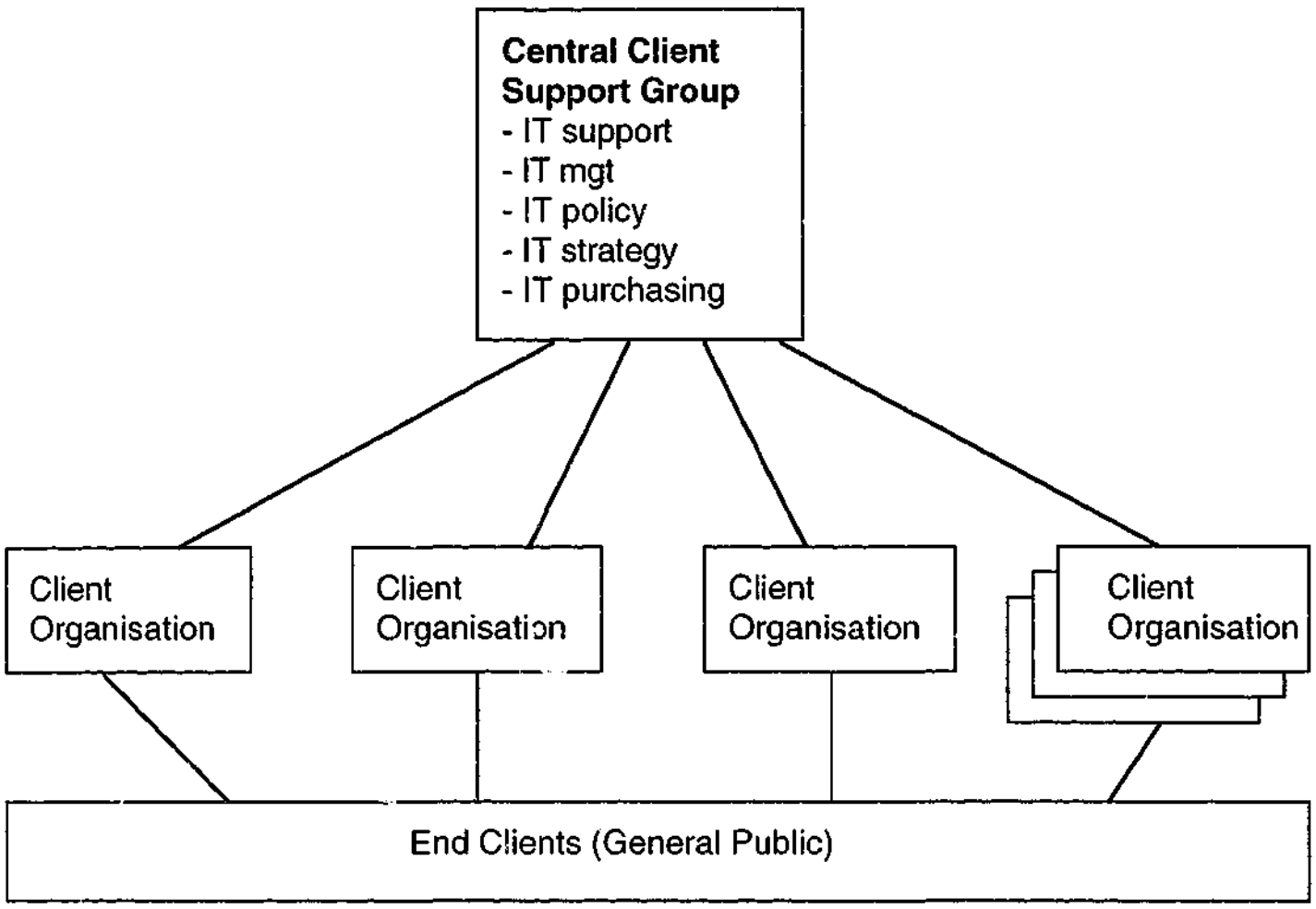


Figure 5.2 the client/user structure

The brief for the analysis group from the client support group was to develop a methodology for specifying requirements for a transaction management system which could be adapted by each of the client organisations according to their specific needs. The project was not about providing the actual requirements specification but about developing a methodology and a set of tools for specifying requirements. The brief did not ask for templates, it asked for a methodology. The concept of a template-based methodology using common transactions grew out of the development process within the analysis group. The implementation development was tendered for separately from the specification methodology. At the time that the methodology and tools were developed the vendor was not known and the tendering process was still in progress. So the platform was not known and platform considerations were not taken into account.

The main aim of the specification methodology was to identify common infrastructure elements or to look for a common infrastructure that could be put into place to deliver on-line services for several different client organisations.

The objectives centred around economy of scale and included:

- improving customer service
- improving accessibility to government services
- lowering the cost of doing business
- developing local industry.

In the course of the methodology development about six client organisations were used who provided real examples of transactions. These were developed into example use cases with a minimum data set that was incorporated from the real examples. These six organisations had all their transactions specified as part of the development.

For the implementation side of the project the central client support group put out a public tender to vendors requesting a common architecture to deliver transactions across various channels. The available delivery mechanisms investigated which provide public access were public kiosks, telephone systems using interactive voice response (IVR), human-assisted call centres (this was a good idea but there was not the funding to explore it), and the Internet. The specification methodology described here was developed for the Internet channel.

The implementation tender was won by a hardware/software vendor consortium that developed a public kiosk system which provided a common

engine for all three channels. This meant that there was a general architecture and different presentation methods.

The end users of the system(s) were the general public who could choose to use the system services or not. Officially the target user was "the average person" so the user interface provided by the software vendor consortium was a very simple design ("push-buttony") – the kiosk kind of look and feel. It was also designed specifically for people who are vision impaired – large letters, high contrast etc.

	Common Transaction
1	Obtain information Information from all organisations
2	Make a Payment Utility bills, Rates, Fines, Vehicle Registration
3	Monitor Progress Check progress of applications and accounts
4	Book a Service Tests, Building Inspections, Venues, Events
5	Provide Feedback Community Action, Comments
6	Buy/Order a Product Reports, Certificates, Permits
7	Change Client Details Name, Home Address, Mailing Address

Table 5.4 The Seven Common Transactions

The analysis group examined how the government agencies were going to specify their requirements for implementation across this infrastructure. The first thing was to identify and scope the transactions and then identify the most common transactions that the agencies were likely to implement. Originally six common transactions were identified and then another giving seven common transactions all together (see Table 5.4).

The vendor consortium implemented service packages for those transactions and the system was designed around those transactions. The transactions

became very important in the specification process because they had a pricing schedule associated with them and variations to a common transaction cost the client organisation extra.

5.4.3 The Development Methodology

In keeping with the consulting organisation's philosophy outlined in section 5.4.1 above the methodology developed for this project was in-house and specific to the objectives of the project. The methodology was based on use cases (Jacobson et al., 1992) and a generic object model using Rumbaugh et al's (Rumbaugh et al., 1991) Object Modelling Technique (OMT) notation. Jacobson's full development methodology was considered too open-ended, with too many decisions for client organisations to make. It was also assumed that there was someone in the client organisation who knew what an object model was – an IT liaison person. *"We give them [the client] tools for their IT person to do a specification with ...not for an end user to sit down and specify with ... and in some cases ... it's just easier for them to hire someone to go in and do it."*

The analysis group could not make the methodology specific to a particular client organisation because every client organisation ran differently. It took some time to arrive at a tool kit and specifications that not only explained the methodology and how to use it but also partially specified the common transactions. The partial specification was presented with a use case methodology i.e. *" ... the standard pathway, the exceptions, maybe an alternative for the generic case - much the way Jacobson might describe [it]".* When ordering a product – an end-client might say *"I want it to be red with blue stripes, I want 6 of them, I want them delivered to my home, I am going to pay by credit card etc."*. That is, most client organisations had some kind of "product" to "sell" so there was a general flow to a transaction. For example, payment is based on order quantity and unit price – i.e. there is a general set of

rules for payment calculation. After looking at variations a general pattern was arrived at for the use case along with documentation and some common data elements. Client organisations were invited (optionally) to supply elements for a minimum data set. The client organisations were given a general pattern for a transaction that could be configured to how the client/end client wanted it. These partial specifications were called "lazy dog" templates. *"There is one general methodology, one [generic] object model and there are a set of seven different templates, for each [common] transaction type that you can use and you can tailor the templates in what I call "lazy dog" templates – i.e. they are half filled out - it is not a blank form."* The concept of "lazy dog" templates is used in engineering specification. There is a standard technical specification which is half filled out and variations are added.

The partial specification, or "lazy dog" template, was presented to the client organisation (specifically to the IT liaison person or team within the client organisation). This template could be configured to the client/end client's specific needs. Other common transaction types could be added at any time. This may have meant adding another object to the model.

5.4.4 Project Documentation

In-house documentation was sighted: blank and filled out templates. These were booklets containing how to use the template, the generic use case, the generic object model, interaction diagram etc. *"We ran workshops to actually detail the transactions with a number of groups and then we reverse engineered..."* to a common pattern and set of common transactions. The methodology evolved slowly from sets of reports.

5.4.5 The Requirements Engineering Process

The application of the specification methodology was based on booklets sent to the client organisations containing instructions on how to use the template, the generic use case (diagram and script), the generic object model, interaction diagram etc. All business rules were required to be documented. The methodology *" ... should be thought of as a methodology for specifying a transaction management system not a full on-line system – so it was a very controlled process."*

The first task for the client was to work through the general use case flow diagram and instructions to see how well their transaction matched the common model. This test is called a "goodness of fit" test. Goodness of fit was important because pricing was based on it. The more variation from the common model the more it cost the client organisation. *"We had no idea when we worked with the specs that they'd be used in that way but it has become very important to actually using them ...we knew there were going to be variations because we were just going for the 80/20 rule, the most likely case."*

Customising the template involved modifying the basic flow diagram (based on the "goodness of fit") and the object model, modifying the use case script by striking out (not removing) elements, so that someone could look across the page and see what had been changed. Few modifications were made to the object model by client organisations. The generic model seemed to apply to most situations.

Elicitation

The IT liaison person within the client organisation sometimes did the knowledge elicitation or requirements gathering based on the template documentation. He/she may have needed some preliminary coaching in

understanding how it all worked. Some client organisations hired outside consultants to do requirements gathering if they were going to hire a contractor to implement the final system anyway. Client organisations received the documentation ahead of time and could choose to consult another client organisation that had already been through the process.

The elicitation process was not seen as specifically object-oriented but this consultant claimed to think in terms of objects at this stage because *"...that's the way I think ...so it's hard for me to unbundle it... we don't say to them [the client] we are really talking about objects and we are using an OO methodology. We just do it and they just want to specify their transactions"*. Another consultant in one of the teams, who was not an object-oriented person, helped one organisation to specify a suite of these transactions. He had a hard time with a couple of concepts. One is that he wanted to *"... see in the model all the data components [which were] really more like patterns than objects. We wanted him to think of ... going through the flow of how someone purchased a product that it fitted within the model of acquiring a product"*. He was thinking about implementing the system not about managing the transactions. *"And I don't think that is particular to OO or not – it's more to do with what the nature of the system is that we are dealing with. But those are the kinds of concepts that I had trouble getting across aside from the terminology – it's a system thing you know"*.

If the consulting organisation was assisting the client or doing all the knowledge elicitation for the client the first contact is upper management to set the scene and scope the project. Then if they already have a data model, documentation, data dictionary, etc on the current system then that can be used as a starting point for the data definition. The basic techniques used are interviewing and working with use cases/scenarios. It is an iterative process. The consultant might go back to the client, on average, three times.

The whole transaction specification would take about a week. Information gathering generally took place on an initial half day and then was followed up with an hour in the next consecutive two days. Probably a week after the start the specification was given back to the client for review and the consultant walked them through it.

Modelling

The generic object model was a standard Rumbaugh et al (Rumbaugh et al., 1991) OMT model and the rest of the methodology was built around Jacobson use cases (Jacobson et al., 1992). The IT liaison person within the client organisation communicated the information from the model back to the client users. The client focus was on the use cases rather than the object model and the model was not explained to the users (because they would probably find it difficult to understand), but just to the IT person. *"We tell them [the users] that the model is technical mumbo jumbo..."*.

The clients fill in the use case template: e.g. product name, ID number, description, pricing information and then they document any exceptions. For example, *"...we don't have an ID number, we've got payment up front, we don't do that etc"*. Elsewhere the client can add notes about business rules or other identified data elements or attributes. The client actually edits the template to customise it to their own version of the common transactions. It is only the IT person who might go through the object model. OMT-style interaction diagrams were used but were put into the appendices and were rarely used with the users. So the users did not see the model, just the structured dialog (of the use case) and the basic flow.

When asked, ***"Is that in general your experience that object type models can't really be shown to users."*** The consultant replied *"Well you know I wouldn't show them a data model either ...the closest I've gotten is working with this type of flow diagram (use case flow diagram)...they can follow that pretty well*

but they don't usually have the patience to really work through the interaction diagrams or the model. It just takes too much explanation."

There was a standard process modelled as a flow diagram and use case and the client was asked to tick a box if there was a goodness of fit. There was one object model that covered the whole transaction. The client organisation, via the IT liaison person, identified the parts of the transaction common object model that applied to their transaction and could black out any unnecessary objects in the model in a similar way to striking out unnecessary parts of the use case script.

The consulting analyst believed that the use case/template approach worked well but *"... you still need the middle man [the IT person]. The overall approach is quite complex and quite rich. It is different in that they [the clients] need to think differently about the fact that they are managing their transactions in terms of the specification."*

Validation

Validation was not formally done on the fine-tuned (resulting) transactions. *"If there is any validation, it's against case files – but it's just a normal walkthrough really. It's not a rigorous cross validation more of a backtracking looking for omissions and inconsistencies. So it is firstly a formal walkthrough and then the client can take away the specification and case files and look at and discuss them. They might come back with things that have been missed."*

5.5 Case 3 A Fault Management System for a Telecommunications Organisation

The consultant in Case 3 used object-oriented methods to specify and build a fault management system for a telecommunications organisation. Models were

based on use case scripts, OMT class models and interaction diagrams although the interaction diagrams were not used much until the design phase. The consulting organisation for Case 3 is the same as the consulting organisation for Case 2. The analyst/consultant is different and the client and project are different.

5.5.1 The Consultant and the Consulting Organisation

The consulting analyst who was interviewed for this case study holds the position of Senior Consultant within the consulting organisation. The consulting organisation as described for Case 2 is a flat organisation in that there are senior consultants, principal consultants and the managing director so the majority of employees are senior consultants. Most senior consultants are self-motivating and are required to bid for projects and then organise and carry out the work.

The consulting analyst had been three and a half years with the company – two and a half on this project. Previous experience included working on an Open User Interface product for nine months, in a small group of people, building network management frameworks for open systems. Before that the consultant worked for Unisys for six years and was involved in software product development including requirements analysis. He also did contract software development for two years.

The consultant considers himself to be a very experienced developer who has spent more time than the average developer in requirements engineering. He has been doing object-oriented systems development for about four years and has delivered formal courses in Object Oriented Analysis and Design through the consulting organisation. When asked to comment on the main advantages of an object-oriented approach to system development he replied: *"The key points to me come from my experience, and this is borne out by the [system]*

project. The ability to evolve code in isolation behind interfaces just seems to be the key benefit ... the ability to make significant changes to significant amounts of code and not destabilise parts of the system which you are not directly working in."

5.5.2 The Client and the Project

The client was a large telecommunications organisation. The project began in-house for the client and then was outsourced to a third party management client acting for the original client.

The project is a fault management system for managing planned and unplanned outages in a transmission network. It was a five-year project and has involved two and a half years of serious development work for this consulting analyst. The project was funded incrementally and for the first stage the deliverables were a suite of requirements and analysis specifications. The requirements model was a use case model and there was also a prototype.

The consultant classified the project as pseudo real time. There are some real time "feeds" into the system. It is real time in the sense that there is a component which manages service interruptions. Once the equipment starts setting off alarms and those alarms are brought to the attention of operators in the work management centres, the operators have to immediately notify all the appropriate parties. That involves raising an exception event within the system and providing as much information as is available to all involved parties. The system had to do some complex reports and they had to come back within 30 seconds to 2 minutes. Further to that, there were some real time "feeds" in the database, which were constantly triggering update transactions.

The hardware platform for the system was UNIX with 4 servers. The database was Objectstore 5. The operating system was HPUX 10.1 with Open UI as the

front-end and it was deployed on HPUX workstations and also Windows NT 4.0.

5.5.3 The Development Methodology

The development team (including members from the consulting organisation and members from the client organisation) was mostly a team of 12 and peaked at a membership of 15. There were two subteams to do a lot of the early work and the consulting analyst supervised one of those subteams. There were project meetings for the entire duration of the project, on Monday mornings, which would run from one to three hours.

The methodology used for system development was an in-house object-oriented method. It was based on other methodologies that members of the team were familiar with. "*We sampled from methodologies that we were familiar with. We used bits of other methodologies as appropriate.... five of the developers had significant experience of building similar systems elsewhere... What that meant was there were three or four people who were able to contribute to a methodology that picked up bits and pieces from a number of influences... They just all brought their biases and their interests and thoughts.*"

The development process was heavily influenced by the people who were available, and the fact that they had come with quite considerable industry experience in this kind of software development.

5.5.4 Project Documentation

There was no proposal or a written outline of the project available. "*The client didn't specify the system that way*". The client employed a group of people to come up with the specification, in this case the consulting organisation.

In-house documentation was considered confidential. There was no available in-house documentation since the consulting analyst was not able to take any documentation away from the project when he left *"I probably couldn't even sketch them out for you now. I have got a general idea of what they looked like and how many there were and how the relationships flowed and so forth."*

5.5.5 The Requirements Engineering Process

Elicitation

Knowledge elicitation and information gathering was done explicitly. There was a small group selected from the user community to be sponsors, who got involved early and stayed on to lead the transition from the development into a field trial and into the production environment. Two of them were in Brisbane, one of them was in Melbourne. There was a team of eight altogether in this user group. They were drawn from other management centres as well. *"There were three guys who we had most of our dealings with and we were free to ring these guys up and discuss things with them. They were always on the end of the phone and they were very helpful and very positive."* One of the main members of the user group (the main business contact) was a network manager with about 25 years experience in transmission management who was one year from retirement. He knew all there was to know about the client's management of their transmission network. He was involved wherever possible and he played a user liaison role and a business expert role. He was called a "subject matter expert" in the client organisation's terminology *"...there was probably nothing you could ask him about transmission or about the business domain, that he could not answer. A lot of the requirements model was drawn by talking to these guys and verified as well through the development phase."*

The consultant felt that the project was a joint application development style of project where the users are so involved that they almost owned the project as much as the developers.

Elicitation started early on and the user group was involved right from the start. The specifications were drawn up after the consulting analyst joined the team and the user group was part of that process.

When asked ***"Do you think object oriented when you are gathering your information or does it become object oriented later on when you start building the requirements?"*** the consultant replied:

"I think you think object oriented right from the start. One of the things that pushes you that way is that you may or may not be prototyping. So in our case we were prototyping and that meant developing a reasonably functional prototype of every view or every screen. Some of the screens were quite complex and so there was a lot to be worked through or to be got right. But they formed the basis of the use cases and also the basis of the first cut of the production graphical interface. And if you're prototyping a graphical interface prototype and working through and developing use cases, you are talking about graphical objects and you naturally extend that and start talking about business objects as well."

Knowledge elicitation was done using interviews with users and the special user group. It was highly iterative to the degree where the subject matter expert would be calling in every couple of days. *"It would have just been a conventional sort of thing, throw some prototype together ... and that can be done very quickly. Get the guy in, sit down and work through our current prototype and that might have happened once a week for twenty weeks."*

Working on this project was one of the subject matter expert's main job responsibilities. He was freed up from some of his network management responsibilities to come and work with the system team. He probably had a day or two a week at some stages to actually come and spend with the team. It involved substantial commitment from the business side. *" ... even after the analysis phase finished and the requirements models and the prototype were delivered, there was still a great deal of contact and a great deal of iterative development and feedback with these guys – all the way through the development. A lot of that wasn't changing the requirements spec, a lot of that was to resolve missing detail in the requirements spec."*

Modelling

Requirements models were based on the textual versions of Jacobson's use case models (Jacobson and Christerson, 1995), OMT object modelling graphical notation (Rumbaugh et al., 1991) and Software Through Pictures (STP) which was the case tool available to the team. Interaction models were used for dynamic modelling. The team saw a need for a static type of model and a dynamic type of model and use cases. The consultant described it this way:

"We saw a need for a static type of model and a dynamic type of model and use cases ... the use case model was more stand alone and really became the functional statement that went behind or reinforced the user requirements prototype ... The interaction models weren't drawn until three months into the development phase, so discount them [as requirements models], the class models were refined considerably to static models, they were refined considerably again, two – three months into the development phase but a pretty good first cut was developed at the end of this analysis phase as a sort of business model. But really most of the first phase of the work developed a use case model and a prototype to go with it. Those were the key vehicles for delivering the requirements."

Software Through Pictures generalised the characteristics of a number of methodologies and allowed simple drawing of diagrams - so that *" ... didn't tie the team down to strictly OMT models"*. It did support OMT in that the STP classes used the Rumbaugh symbols and all the symbols were consistent with Rumbaugh. But it was a fairly generic kind of object interaction graphic editor. STP was used because of its availability - *"I think we would have done static class models, we would have done those regardless of how we would have drawn them. We would have done object interaction models regardless of what we had available to draw them as well."*

Use case models were used extensively. There were approximately 80 use cases, which were expressed in tables on an A4 page in 10 or 11 point font. They were quite extensive and no use case was less than three-quarters of a page. Some went for 2-3 pages.

When asked: ***"...did you do modelling sequentially that is, after the information gathering, or did you sketch out models whilst you are gathering information?"*** the consultant replied: *"There was a very definite distinction between the first class model and its completion and delivery and then we all got to work on interaction models as a way of refining the class model. The real sort of main deliverable in terms of the class model is the business model and it was really a first cut class model from which the application schema would be cast [from the use cases].*

I would say that [the class model] was drawn up quickly within the space of probably a week to two weeks. But I would say that there were fragments of that model getting developed in a couple of people's heads for probably three months beforehand. The development of that model was not done publicly, or the first cut of it, so after that it was tossed to the team and it just diverged. The development of that model was done as these discussions were going on. As the requirements were being collected as the requirements modelling was being

done. But I would say that it was being done largely privately and it was not written down until the last minute when it was just a dump."

Further questioning on mental modelling followed: ***"How and when do these mental models start forming inside your mind. I mean, you have said that this is what happens and at some point it gets turned into hard copy or whatever, do you think in general people doing this kind of work are mulling around mental models in their head?"***

"Firstly, when we talked about requirements and collecting requirements and putting them into the requirements model, we talked about thinking of objects, and someone needs to be identified, there will be someone or there will be people whose responsibility it will be a little bit down the track to start casting bits of models together and someone will be tasked with that. They will be thinking about a business model at that stage. So as they go through they will be listening to discussions and working on the requirements model and they may or may not be writing things down, but this is my view and this is how I work in this situation. You will be listening very carefully and collecting and cataloguing constraints and refining the abstractions in your mind."

This information that is used to build the mental models would come from the group of users and from project meetings and the general requirements gathering process and activities. There was an element of the client or user contributing to the mental model of the object model and then the refinements, the more technical aspects, are done later on with discussion with other members of the analysis team.

Some more comments on mental modelling from the consultant:

"You see what you will come up with will be an abstraction in your own mind, which you probably cannot fully express, but you might feel you can express it

but you wouldn't want to. For me anyway it's a mistake to try and rush in and write that down and stick it into a case model. Because you know its going to change so you shouldn't do that until you have resolved enough, not all, but enough of the question marks in your mind as to what that needs to look like. So you might carry round an event or an account or a customer object or something, you carry around a picture of how that is shaping in your mind. Someone in a meeting or a discussion will say, 'Of course, you know we only ever had one of these, and that will change' and you can say Ah! Test that against my understanding of what a customer, or event or a facility or whatever the abstraction is going to be and that might either verify or it might contradict it. If it verifies it you probably let things go and move on to the next point. If it contradicts it, you need to pick it up and mine that and get to the bottom of that."

In summary, the requirements specification process started with requirements models based on the textual use cases. At some point, there come the class definitions and the object model (the OMT static model) and then later on, the interaction models.

A further exploration of the use of use cases, particularly any limitations, led to the following comments:

"... for instance, the requirement that it will initially support 20 users but one day it may scale to 200. That's a requirement that should come out of your requirements modelling, hopefully there is a use case, there's something somewhere in a use case that captures that kind of thing although I feel that use cases if used in isolation with no other requirements modelling can miss some of these system wide axioms."

"I actually think they [use cases] are useful, you have got to do them as a mechanism of exercising your requirements, your understanding of your

requirements, exercising the business model and even going further and exercising your design model [and understanding it] ...but it is dangerous because you can go through create a patient or bill a customer use case a thousand times and never hear the exceptions or never hear bits that you've missed."

So, the three main models produced were use case models, the static class models, and the dynamic interaction models. There was a prototype as well which included the use case model. The use case model was categorised as a dynamic requirements model, a functional model. In this case the group of users only ever dealt with use case type models - they never had to understand the OMT model or interaction models. The object model and interaction model are models only used within the analysis team and understood by members of the team. This idea is illustrated by the following exchange:

Would you, at any point, have shown this group [the user group] ... object models or use case? *Yes.* ***Did they understand how use cases worked and so on?*** *Yes.* ***And what about the OMT model or the interaction model?*** *We would have stopped short there.* ***Is this because you do not think that the users/clients would be able to understand the OMT models (say without extensive explanation or training),*** *... Yes. Unless the 'users' were IT-literate people, which most aren't.* ***... or is it because you have tried showing these types of models to users/clients in the past and they haven't understood them?*** *I don't think I have ever tried, at least not with real business users. I presented an Object Modelling Workshop for several years, and I can assure you that it takes a surprising number of IT people several days to understand the basics of conceptual modelling (class versus instance, relationships).* ***Do you believe it is not necessary to show them?*** *I believe it is not only not necessary, but potentially dangerous. It is the analyst's job to perform the use case to business object model translation.*

Validation

Validation of requirements was based on revisiting the use cases with the users and the prototype, particularly the subject matter expert and user group *"So we expected each other, we expected them, to pick up errors and omissions."* No formal validation models or processes were used *"... there was a framework in which this use case model sat, so there was the high level project flow framework if you like, there was no other validation model designed in the sense that we had a requirements model, we had a prototype."*

There was a separate acceptance test suite developed by the users, but that was not set up until well into the development stage approximately six months before acceptance testing was due to start. *"That was completely separate from our use case model. All the way through we used our use case model to test things as we were developing as we were showing users."* So the test suite was used for testing the final implementation not validating the requirements specification.

Validating the specification was done mostly by walkthrough based on the use case models and the prototype.

"There was no formal validation. There was an over reliance on the use cases, so if anyone had said, and I am sure it was signed off by these guys at the time, 'Is this a complete and accurate requirements model?' they would have said 'Sure, we use the prototype everyday, we showed it to all our colleagues, looks pretty good'. They produced 5 inches of documentation and we worked through enough of their use cases so that we think they are pretty accurate and at the right level of detail. No-one would have gone through it from start to end, and no one would have separately tried to create a validation model or validate according to the criteria."

5.6 Case 4: A Generic Insurance Package

The consultant in Case 4 had extensive experience in using object-oriented methods to specify and build actuarial and insurance systems. The models which were shown to users were based on adhoc diagrams, rich pictures and screen simulations rather than OMT class models or interaction diagrams, although the diagrams based on more formal notations were used within the team and in the design phase.

5.6.1 The Consultant and the Consulting Organisation

The consulting organisation is a small group of three people. The participating consultant is a director and a partner. The members of the consulting organisation work as consultants in the object-oriented field but are also mathematicians " ... *with a particular view on life*".

The consulting analyst has been with the consulting organisation for just on a year and has been doing requirements engineering for about 22 years. He originally managed an operations research department where the group was formally charged with large-scale software development, solving organisational problems and often built software as solutions to those problems and therefore had to develop users' requirements.

The consultant's background is in mathematical modelling and he believes that he has been doing object-oriented analysis longer than anyone else in Australia. He also believes that he was one of the first commercial users of object-oriented systems, SmallTalk, in 1985. Although the consultant has not taken any formal courses in object-oriented systems development, he has delivered them including the first course in Australia. He trained as a mathematician then did a postgraduate diploma in computer science in 1980.

He came to object-oriented systems when working in the operations research department where he was a member of a group which built custom made software tools for users. The group developed specialised languages for insurance problems and built a compiler for an actuarial notation language and another specialised compiler for a language called the 'benefit definition language'. Both of those projects were extremely successful, one of them still delivers a commercial advantage to the organisation. The other did for about a decade.

5.6.2 The Client and Project

The project discussed here is a receipting system. It is a subproject in a much larger project where the consulting organisation is building a particular class of administration system for insurance applications. The consultant came to work on this project because his colleagues had already worked there and the consultants were invited to work on the project because of a personal contact.

The client is a commercial software developer that builds generic packages and then sells them to clients in the financial sector. This system is meant to be a first stage. The organisation believes that it needs to move into object-oriented systems. The consultant was engaged " ... *not so much as a requirements analyst but more as an OO mentor. Now in that role one of the first things we did was requirements and then moved on to design and so forth.*"

The objective of the project is to produce a receipting component for a larger insurance system and for it to be a commercially saleable product that can stand-alone. There are also two other objectives. One is to take a group of the best programmers in the organisation and move them from conventional to object-oriented systems development (a training aspect). The other is to get some experience in producing a larger, more comprehensive administration system from other component systems. That is, a modular approach where the saleable components will be built in a modular fashion.

The client organisation has a very high retention rate. It is considered to be a good environment in which to work and the staff is highly motivated. The people who work in the client organisation tend to stay on for 5 years or more *" ... it's part of their thinking, moving into the new world and everyone's keen to... The project I'm on now is the plum project, everybody wants to work on it and it's only the chosen few who are."*

The need to move into object-oriented development was driven by the expectations of clients and commercial demands. The client already had some very successful products in the marketplace but " ... *they are aware that these products are aging and the presentation layer is fairly old-fashioned and they know that they need now to handle a wider range of products and provide extra facilities and many of their clients have been talking about object-oriented architectures and they want to conform to that."*

The project is a transaction-based one which started about a year before this interview and there was a first release due at the time of interview. The consultant believes that it has been well put together although there were some technical problems with the database. Otherwise the system is ready for limited release. The twelve month time frame is *"... considerably longer than they would have spent if they had been using conventional technology rather than moving to OO. They are using a completely new technology."*

The project is targetted towards a range of hardware and operating systems and a range of databases - it is a generic portable system. This portability has also added to the length of the development time frame. Operating systems are standard mainframe operating systems particularly IBM and their style –AS400 machines, UNIX boxes, Windows, etc. Databases include Sybase, Oracle etc *"... relational databases because that's what the clients want."*

As a packaged system the number of users could be quite large because of the different types of processing since the targets are financial clients *" ... well let me*

give you an example... [potential clients] like [a motorists association] will have a portfolio of business services which includes insurance – will require a relatively small, a modest number of operators to do that processing whereas if you go to one of the large banks or one of the large financial institutions you might have 5 times that processing. So it's meant to be a scalable system and it's meant to be ... to accommodate large numbers of users especially the final administration system."

There are relatively informal project meetings for the team. They occur a minimum of monthly but usually more regularly than that on a need to know basis. There are actually two projects going on simultaneously – one is the development of this system and the other is a parallel infrastructure project. The key piece of infrastructure has been built and is maintained by the consultant's partner. So the project management has to be for the individual modules and the larger infrastructure project.

5.6.3 The Development Methodology

The consultant does not use any specific methodology in this project or any other projects. In his position before his current one he worked for an organisation which was *" ... not aligned with a particular methodology though one came across methodologies all the time so one used those techniques in various ways. ... I haven't been, let's say, an advocate of any particular methodology from start to finish ... See I don't believe in methods as such ... What I talk about is a underlying concept rather than a methodology. A methodology seeks to impose a concept – when it's used badly it certainly does – I think methodologies and parts of methodologies are useful but they're just props and tools and can be picked up and thrown away as required ... I think a methodology is only as good as the deep understanding that people have of the concepts that it's built on ... a methodology is no good on its own.... you need to*

have rigour and the diagramming notations and the steps in the methodologies give you that but you also need to play in the sandpit"

The consultant sees object-oriented modelling as *" ... a superb way of modelling the real world"* which allows a high level of abstraction *" ... I'm unlearning some of my [data modelling] prejudices being back with [my old colleagues] and starting to look at objects and classes more in terms of services than as data and deferring the internal structure later and later and later into the design."*

In this project the team has been fairly rigorously applying the use case concept and traceability comes from the use cases. The team started with the requirements, moved onto design, and then became involved in other parts of the development. " *... the single thread through the whole thing has been the use cases and a lot of the objects are still there and they ...well they've got the same name but the way they're organised is quite different."*

Prototyping is also an important aspect of this project *" ... at [my former firm] I didn't have the luxury of being able to cut code. I tended to give advice and build models in the abstract and they'd be very elegant in the abstract but until you try it you don't know whether it works ... to test even your requirements let alone your design is to actually implement something and watch it fall over – tear up the paper and start again. So again this is a lesson I'm relearning. That in the quantitative research dept ... I joined in 1977 and I was prototyping in the first couple of weeks and I've never stopped."*

5.6.4 Project Documentation

There was no documentation available at all for this project since it was considered confidential.

5.6.5 The Requirements Engineering Process

Requirements engineering in this project where the same team is undertaking the whole development process is seen as ongoing even at the final testing stage. *" ... it doesn't end either. I mean in a way now that we are testing we are still gathering our requirements – not so much the business requirements but... a rather vague requirement is being heavily reinforced when we've actually got to use what we've produced."*

Elicitation

Elicitation is done explicitly where possible but because this project is not an in-house development the users are not captive to the organisation. It is very difficult to get user involvement until the package is complete. There has been quite an effort to gather user requirements but it has been quite unsuccessful. So, for the purposes of elicitation, the users are business analysts and pre-sales people in the client organisation and that means that there is a question mark over the validity of the information.

Prototyping, particularly using illustrative methods and tools like PowerPoint slides to mimic input screens, is seen as enhancing requirements gathering and later acceptance of the requirements *"If you get, as part of the requirements gathering, a prototype you get much better sense of requirements. One of the things I've done on this project which I've actually done before is I've used PowerPoint before we had an interface to simulate an interface and the thing is that it's not just having a picture of a screen, you can run a slide show and see how you interact with the screen and I actually threw away the text use cases when I got to the design phase and just did it all that way. The interface developers were using that as a guide."*

When asked, ***"Do you think "object-oriented" when you are gathering user requirements? That is, are you thinking in terms of identifying key objects at that stage or building mental models of the system? Or does the object-oriented nature of the system and the models grow later during formal modelling and design?"*** he replied *"I think that I do immediately start thinking of key objects during requirements gathering, not in any formal way, they just pop into one's head. I don't agree with the implication ...that identifying objects and 'building mental models of the system' are mutually exclusive. One can help the other."*

When asked whether he is explicitly conscious of implementation details when doing requirements specification he replied *" Yes ...you can't avoid it. And that's really another sort of aspect of having objects in the use cases that you can zoom up and down. You can simultaneously be doing implementation, design and requirements. The cost of going back is much less than it is in the conventional method where you have to model the machine rather than model the real world."*

Modelling

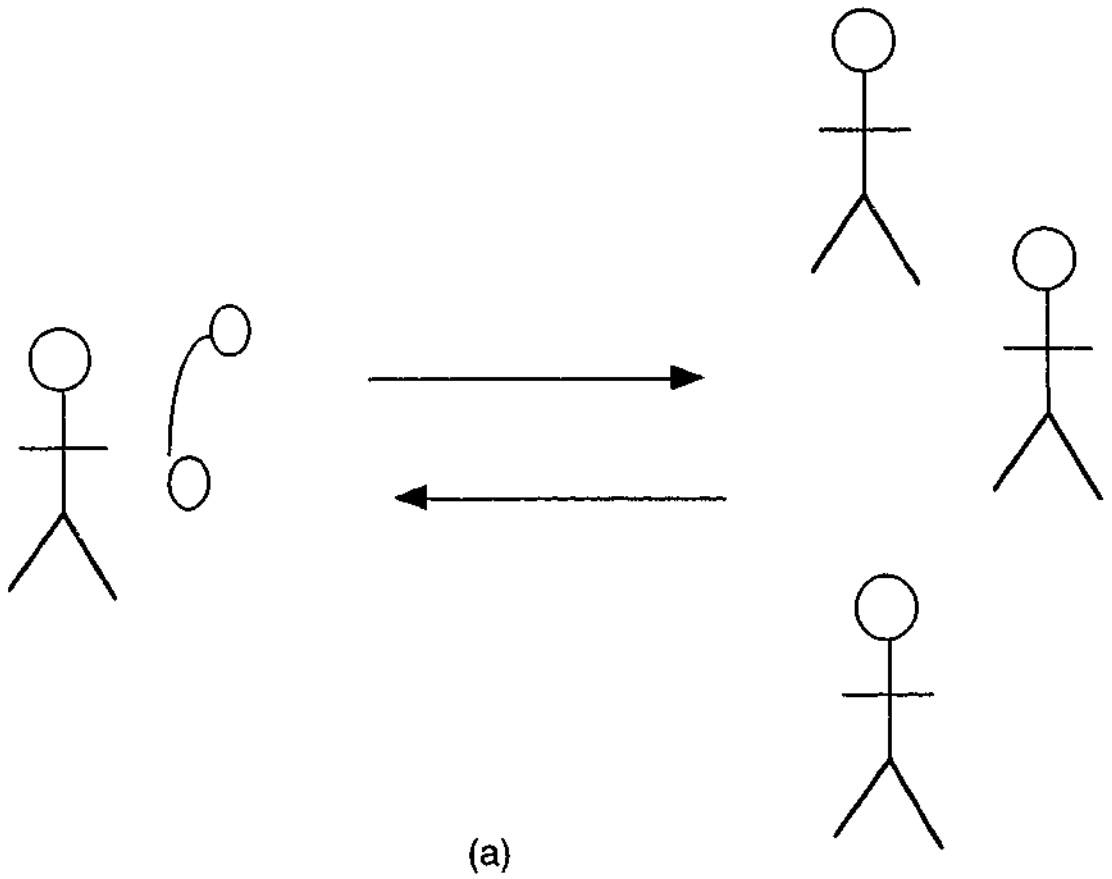
In this project, when the consultant arrived a requirements document had already been started and there was an entity relationship diagram and some processes as well. *" ...and we just circled around those two things over and over again refining and refining and arguing about how things should be expressed."*

Both entity relationship modelling and the UML notation were used to produce requirements models as precursors to design and implementation models. There is a perception of two categories of models, static models and dynamic models, or class models and state-transition type interaction models. *"I've used Booch, that is Booch pre 'the marriage'. I don't think there's ...these notations only differ in the fine detail ... and when you actually flipping between the design and the implementation they are really good anchor points*

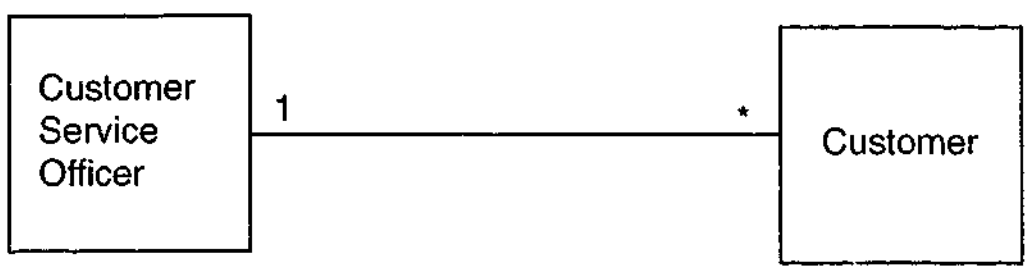
provided you don't go bananas about state-transition diagrams and everything."

A discussion about how the consultant went about initial modelling of requirements led to the following illustrative comment about the usefulness and necessity of ad hoc or informal models " *...and in every project I've ever worked on be it a mathematical project or software development project there's been a few key pictures. The one I'm working on at the moment is the billing cycle – it's a wheel and its got the steps in the billing cycle on it and that's in everybody's head and everybody talks in those terms and it's just the key base thing – it's the conceptual core of the thing ... I'm a great believer in ad hoc diagrams that give the picture that springs from your understanding of the problem and in a lot of OO work the process of development hinges on one or two of these pictures. ...[and] the trouble with that [using ad hoc diagrams] as a methodology is that its difficulty is that you can't capture it, you can't describe it in some way that anyone [else] can really use it and that's precisely its strength because it handles those parts of the things that don't fit in the normal descriptions and every project's got an aspect like that."*

Further discussion about how the consultant would model real world situations for users led to a description of his use of 'rich pictures'. These rich pictures are similar to the rich pictures in the soft systems methodology (Checkland and Scholes, 1990) but they are used by the consultant in a more simplistic manner. At this point in the interview he drew an example of a rich picture that he would show the users and the corresponding OMT fragment that he would not show the users. This example is in Figure 5.3.



(a)



(b)

Figure 5.3a) Rich picture diagram for users (b) corresponding OMT fragment not shown to users

"I would have tended to build those rich pictures very early on and fed them back to them and adapted a description of the requirements." This informal modelling using rich pictures is based on artifacts that belong to the system itself, that is, a specific thing that belongs to that system and is not in any formal methodology. *"Apparently that strikes a chord with the user too. I mean if you draw a picture and that doesn't make any sense to them then you draw another one. But once they start talking back to you... part of the problem is evolving a*

common terminology that then packages concepts up and you agree on that and then you keep going. That's what you are seeking to do. So what I ...you have to be flexible."

In addition to ad hoc diagrams and rich pictures the other main vehicle for developing requirements with the users were use cases *"Well, according to the theory they are the backbone – you start with them and they go right through to the testing and so forth and I think that's reasonable. Generally speaking I think it's a good way to use them."*

Use case types of models together with rich pictures are the models that were used to communicate ideas to the users, that is, to people who are not computing professionals. Full blown class models or interaction diagrams are not shown to the user, only the informal use case, rich picture version. When asked the following clarification question: ***"What I'm interested in is how you communicate your specification back to the users, an OO specification, how it's actually communicated back to people who are not trained in OO modelling. That is, how do you explain your specification to your clients?"*** he replied *"A requirements specification has to be in the terms that they [the users] understand and those three mechanisms we've already mentioned are the way. The use case, the ad hoc diagrams, and dynamic screen simulations. And encompassing text too, you don't write your requirements document in use case speak from start to finish. You talk about the general context and background. I also use context diagrams as well ...like there's a box and there's what's in the system and how it reacts to other things outside the system – just simple stuff but that's helpful too."*

The following exchange was used to clarify the issue of showing models to the user: ***"When you described your use of various models you said that you would show the users/clients an ad hoc diagram, a prototype screen using PowerPoint or a use case as a way of explaining or describing the requirements but you would not show them an OMT/UML type class or interaction model. Is this***

because you do not think that the users/clients would be able to understand the OMT models (say without extensive explanation or training)?

Yes, but based on experience...

or is it because you have tried showing these types of models to users/clients in the past and they haven't understood them?"

I have tried showing these models to users and it hasn't worked well because many people find such abstractions hard to relate to. My former colleagues at [X] found the same thing with data models. One consultant tried replacing the entity boxe with evocative pictures and achieved better communication.

I am not saying one should do away with the formalisms. They are a powerful aid to one's own understanding and analysis but they are not a good tool for feeding requirements back to the users. It's much better to bend the formalism to the user than the user to the formalism."

Validation

The consultant agrees that it is necessary to validate the specification once the models or the specification have been produced. Validation was undertaken based on walkthroughs (role playing based on use cases) and the *prototype " ... use your prototype – early – then you've really got something that you can feed back. And the other way is to take all the use cases, all of them and throw them away and talk about what you do – a sort of role playing..."*

The consultant sees the role of validation as being ongoing throughout a systems development project and highly dependent on continuous feedback *"I think it is a continuous process. Really when you go into requirements somebody is trying to explain to you an idea or describe to you something they*

do. Now you've got to feed back your understanding of what they are telling you on a continuous basis and then you might decide that they are talking about direct debit processing and receipts and you talk about whether the tape goes to the individual banks or the clearing house and then they tell you something that they haven't told you before. You keep asking questions and you get to a point where you say 'I think I understand it now let's go through it again'. And then later on you'll get to the same point if you like for the whole system and you'll say 'Alright now let's go through the whole system again' and you'll go over some of the ground you went over before as part of the process. So you are constantly doing that. But you do it in small steps and you do it at obvious milestones and you do it for the whole thing at the end."

No acceptance testing was done or planned for this project *"I've not done it, no. Maybe one should."* The use cases together with the prototype were seen as important validation tools. The use cases are "alive" from right at the very beginning and they are used at the end to revisit the specification " *... and say this is what we think we've got, this is what we are going to build ... I wouldn't assert that when you build your prototype you take a particular use case and implement it at some level. I think you implement the system at some level and that would give you a capacity to traverse it using some use case."* In this system prototyping was not an implementation of use cases. Use cases were used to focus on a critical class or a critical cluster of classes for the prototype. Use cases on their own are not seen as adequate for defining or validating requirements.

Requirements were revisited right through the design and implementation because there are always ambiguities, inconsistencies and omissions which did not show up until prototyping and testing *"**Do you usually expect to revisit or revise the specification during design and implementation due to unforeseen omissions and ambiguities?** ... Bet your boots!"*

This case revealed the use of pictures, adhoc diagrams and PowerPoint simulations for discussing the requirements with users. This idea is followed up in subsequent cases.

5.7 Case 5: A Life Insurance System

The consultant in Case 5 professed to using an object-oriented approach to specifying and building a life insurance system but claimed not to use any models either based on use cases or diagrams of specific modelling notations. The consulting organisation for Case 5 is the same as the consulting organisation for Case 4. The analyst/consultant is different and the client and project are different.

5.7.1 The Consultant and the Consulting Organisation

The consulting organisation for Case 5 is the same as the consulting organisation for Case 4. The analyst/consultant is different (another partner) and the client and project are different. This case study is less detailed than the others are because there was a technical problem with the tape recorder at the interview. Unknown to the researcher, the tape was not recording even though it appeared to be. The data is based on an email interview based on the interview script, notes, recollection by the researcher and follow up questions.

The consultant interviewed for this project describes himself as a director and consultant. He has been with the organisation for abou. two years and has been doing requirements engineering/systems analysis for approximately fourteen years.

The consultant has never undertaken any formal course in requirements engineering or object-oriented systems. When asked about what methods he used he replied *"The method I always use is to THINK about the specific*

problem from first principles, this applies equally to business-domain and technical aspects of the problem."

Prototyping can play an important part in the consultant's system development approach. *"Prototyping (in terms of screens to play with) is valuable to support client/development understanding and strengthen sponsorship. Otherwise it is an unhelpful diversion of development effort. Prototyping in terms of getting something running and then changing it is what we always do."*

When asked about the object-oriented concept that an object remains the same from analysis right through design to implementation as being a strength of object-oriented methods he replied *"Objects are only stable near the end. I've never seen ANYONE get an object right first time."*

5.7.2 The Client and The Project

The project is a life insurance system where the objectives are to provide software to implement all the systems required to support a life-insurance business. The project was initiated directly by the client which is the Australian arm of a large international insurance organisation.

It is a transaction-based system with an initial phase of twelve months. If successful, the project may become ongoing over several years. It runs on Sun/Solaris servers, Intel boxes with Windows NT frontends. The number of users will initially be 20 - 50. If the project becomes ongoing and is globally deployed it is envisaged that there will be *" ... thousands of users not counting Web-accessible components."* The project team is a team of seven who has weekly *" ... ad hoc sit around and say what we're doing meetings."* The consultant is not currently working on any other projects.

5.7.3 The Development Methodology

There was no specific methodology being used on this project. The specification was text-based and grew from the requirements gathering process *"Ad hoc notes from whiteboards onto loose-leaf and/or bound notebooks. Typed up 'processed' versions of these in an adhoc, as needed, fashion. Many ideas get 'documented' in the code (use of Javadoc to include design ideas in code)."* This approach was used because *"... it works well"*.

Requirements specification is seen as a process that is always being revisited. *"Ongoing throughout project -- it's the development team's responsibility to make the final product satisfy the business, and as a means to this end, ongoing consultation, discussion, and sessions at the pub are very important."*

5.7.4 Project Documentation

There was no relevant in-house documentation available *"Each team member has a range of books they find useful. Some are more bookish than others."*

Documents available from the client were used as part of requirements specification *"... the artifacts that they use in their business - internal memos, instructions, reports, also spreadsheets etc. do all the requirements documentation."*

5.7.5 The Requirements Engineering Process

Elicitation

When gathering and specifying requirements the consultant said he was definitely not conscious of implementation details and that the specification and modelling process was completely implementation independent. The

general approach to requirements gathering is illustrated by the following comment *"When talking to businces, we have to BE good business people. Being a good requirements gatherer and analyst requires the skill to be able to think in the same way as a wide range of business people and being good enough at understanding the business so you build trust and rapport. The only way to get good requirements is to get the business representatives to feel that you understand their problem and are bright enough and competent enough to be able to solve them."*

Knowledge elicitation was explicitly undertaken and started early *"Yes. Lots of talking early, but ongoing through project."* The elicitation process involves senior members of the team *"Senior developers run the analysis. Junior developers do more leg-work in requirements gathering."*

When asked ***"Is requirements gathering seen as specifically object-oriented, i.e. do you think "OO" from the start? "*** he commented *"No. Think 'business' all the way. (Compare OO thinking to 'Microsoft Word thinking' -- it's only when we come to tidy up and express ideas in Word that we think -- oh I'll try this as a list of bullet points, yes that looks right...) 'OO thinking' sounds weird ... it's really all about just thinking what the best thing to do is with the particular tools at your disposal."*

Modelling

This consultant professed to using no specific methodology or notation and doing no modelling at all since he found it unnecessary. When questioned about whether he did any mental modelling (following up from cases 3 and 4) he was non-committal *"This is really about how people think. Some of us more conscious of the models, others not."*

Neither did he use any informal models - pictures or diagrams to communicate the specification with the users *"No. The users are business people."* or any use cases *"Not really. Use cases are good for focussing attention for a short time up front on the whole range of actors... that's about it."*

Knowledge elicitation was seen as iterative. The team went back to the users several times where some piece of information triggered the need to explore some new feature or aspect.

As stated above this consultant professed to do no modelling at all. His emphasis in conversation was on abstraction. He put it succinctly in the following exchange, *"**When does formal modelling begin**? It doesn't."* Since there was no use of a method or notation the consultant seemed averse to using the terms model or modelling until he redefined them. He saw a model as a set of concepts *"What we have to build is a system that embodies a set of concepts. These concepts are the model. In order to build the system well (so the concepts are clear and clean) it is important for the team to have a clear, unified view of the set of concepts (the model) and so far as the team needs it we use artifacts such as Microsoft Word documents to describe the model. What we are always doing is trying to get a better understanding of the problem domain and our own implementation of that domain. Understanding is often improved via abstraction (= metaphorical thinking)..."*

The models or descriptions that were produced were *"Very very few. On an entirely ad hoc basis and were produced with tools like Microsoft Word and Javadoc,..."* and were only for the team not the users or clients as illustrated by the following exchange: "***Who uses them [the models]**? The team for their own purposes. **Who are they produced for?** The team **Which models, if any, are shown to the user?** None ... the user gets much more 'sales-oriented' presentations."*

Use cases were only used sparingly in the specification process *"... good for the focus on actors, generally too cumbersome, inexpressive, and often dangerous because of the heavy investment in particulars soaks up time needed for abstraction and reasoning about the business."* And consequently there was no perceived relationship between use cases and any more formal static and dynamic models.

Validation

Explicit validation of requirements was viewed with some suspicion *"Validation is often a way of attempting to absolve the development team from truly owning the responsibility for the nature of the product they deliver. It is the development team's onus to make sure they are building the right thing. Early validation does not mean that the thing will turn out to be right or useful, although it can avoid a certain class of disaster. We are always worried about building the right thing. We have to be advocates for our ideas and test them out in discussion with the business."*

Acceptance testing was advocated as useful but use cases and prototypes were not seen as playing an important part in the testing or validation of the requirements. Regarding the finalising of the requirements engineering process the following exchange is indicative *"**Who is involved in the validation process?** The development team and the business representatives. **When is the validation process considered to be complete?** Never? **Do you usually expect to revise or revisit the specification during design and implementation due to unforeseen omissions, ambiguities etc?** Yes!!!!"*

5.8 Case 6 A Stockbroking System

The consultant in Case 6 was a senior project manager for a software development organisation which develops custom-built systems for individual

clients and generic packaged software systems for the stockbroking industry. The consultant was experienced in many methods, both object-oriented and non object-oriented, for specifying and building business systems. In this project a generic stockbroking package was being developed using an in-house object-oriented methodology. Models shown to users were based on prototypes, screen simulations and animations with use case models used mainly at the validation phase.

5.8.1 The Consultant and the Consulting Organisation

The consultant's official title is Technical Development manager. All system administrators report to him and he also acts as a system architect from a software perspective and so is responsible for all designs and all analysis of the software that the organisation develops. He also fills the role of general troubleshooter within system development and assists as an additional resource on the business analysis side.

The consultant has been involved in object-oriented systems for about five years and has worked for the organisation for twelve and a half years. He has spent all of that time doing systems analysis and requirements engineering. Although he has not done any formal training in object-oriented systems he was involved in an in-house course given by a lecturer from Swinburne University of Technology who gave an object-oriented analysis and design course.

The consultant had used non object-oriented methods before moving into object-oriented based systems. He used a relational database management systems (RDBMS) approach based on INGRES and before that he developed COBOL-based systems using traditional structured techniques. Data Flow Diagrams and Structured Systems Analysis and Design Method (SSADM) techniques were used in the RDBMS approach. He believes that although

object-oriented approaches have certain advantages they also have some limitations for developing systems *"It [the object-oriented approach] has some advantages in some of its approaches and the encapsulation concepts work well, however, in many regards we were already doing that even back in the COBOL days by using ... proper use of subprograms and reusable code and ...modular design ... The problem with using DFDs and that is that the models were far too data-centric which was fine if you were doing a lot of retrieval but to do good transaction processing was quite awkward and you really did need very high levels of expertise to get it right in the RDBMS world. That is much less so [in OO] and therefore you can actually end up with much simpler solutions with OO techniques as long as you keep the propeller head so to speak away, you actually end up with systems which are very easy to understand and easy to maintain and that's the big plus."*

He disagreed with the notion that one of the strengths of the object-oriented approach was the delaying of design decisions, that it made for much simpler structures. *"Its interesting because to some extent I would disagree with that. I find that some of the design decisions you have to make, you have to make much earlier and you have to pay more attention to the design because if you get that wrong then everything you do afterwards will simply not work, whereas in the old forms particularly the RDBMS that was certainly not true. ... One of the problems that I find with large OO systems, and effectively what we build here are very large OO systems, is that unless you pay an awful lot of attention very early on to the design and particularly [to] what other objects are being involved – who wrote them and how they are used – you can end up with a much larger mess."*

Regarding effective traceability being available in object-oriented systems because an object remains the same from the time it is identified at the beginning of requirements specification through to the design into the implementation phase he also disagreed with the common wisdom and the

literature " *...my question would be what do you mean by "stays the same". I mean invariably I've found that the base, or perhaps the core of an object may remain very similar. Often what happens is if you are using any form of deep hierarchy, the inheritance trees will change ... recently we've just made a major change to the inheritance model and the particular part of the object structure. Now, on that basis, no the object has not remained the same – it has actually just gone through a major transformation. And a lot of that is because of the way you work in OO – you tend to identify a particular problem domain and you model through it and come out with your object structure. ... And to integrate it best you actually need to change your inheritance model. ...and there are a lot of changes back to the original code because of the domino effect."*

5.8.2 The Client and the Project

The consultant is currently working on several projects. The project which he is spending the majority of his time on is a new flagship software package being developed in Australia. It is a stockbroking package, a back office system for stockbrokers which is configurable to individual client needs. It is neither an off the shelf package nor a one-off to each client so there is a set of core requirements. Assistance is provided by the organisation to clients in customisation and ongoing support.

The project arose as both an in-house project and as a response to an identified need to update an existing product already in the marketplace. It was commenced as a joint development with a major stockbroker. The stockbroking client was in the market for a new system which they put out for tender. The organisation responded and based the development on some software that already existed and then worked with the client in developing the new package. The client's long term objectives were almost identical to the consulting organisation's and so they worked together fairly closely on producing the original version of the software. The software is not customised

towards the client's solution but takes a broader view of general stockbroking package software.

The project is being developed from scratch. *"While we have existing products and we obviously gained a great deal of product knowledge, leverage from that and that also helped on our analysis side because we knew what we were ...we knew the business that we were writing the software for. Effectively we started from a clean slate."*

As expected it is a transaction-based system and the project is ongoing. The initial release of the product is already in production outside Australia. The initial development was in Hong Kong where the initial client was based and they wanted an international solution. Currently the Australian version is being developed and they are also working on UK versions. The software is built primarily with Windows NT clients in mind. There is also some background processing which will run on any NT platform but is recommended to run on UNIX platforms. The main database servers could be on NT but are more often on large UNIX processors predominantly Sun Microsystems.

The number of users depends on the client. For the Australian client there are 350 users. The operation in Hong Kong has in the vicinity of 200 plus users. The product could go global via the Internet and then there could be potentially thousands of users.

The number of people in the team has varied. At one stage the Hong Kong office had about 30 people and there were 15 or so in Australia and another 10 or 15 in the UK. The numbers in the various countries fluctuated depending on the workload. Since the initial set up the numbers in Hong Kong have dropped. *"At the moment, Hong Kong has just dropped ...I'm just organising an additional resources for them. The UK office has about probably 30 plus now*

because we are leading up to putting in a number of UK clients, so there's a big emphasis there on completing the UK work so they can go to production." The development team here has a weekly meeting that is scheduled at the start of every week. *"It's really aimed at being a general catch up about what people have been doing so everybody on the team knows basically what's going on. Also they raise issues of what's coming up. Are there any major deadlines coming up? How do we approach those? And lastly the issue of are there any problems that we don't already know about?"*

The consultant is also working on adhoc projects, which are based loosely around the stockbroking projects, and is also involved in a project with a major automobile manufacturer in New Zealand.

5.8.3 The Development Methodology

The methodology is an in-house methodology based on UML notation but not the complete Rational development method. *" ...there may an ITT (invitation to tender) or something of that nature which we start off with ...out of that document we do business requirements. We have a business rules document. And from that we go to a top level design, detailed design and then again from that there are a number of different levels of testing which are to be put in place through integration testing, system testing, and then user acceptance testing."* Prototyping in the form of a GUI prototype for the users is used in the project. *"We actually do a prototype and then work through the users with that and then gain sign off at that level."*

The consultant believes that prototypes are not always appropriate and can cause problems. *"Invariably prototypes are not thrown away. In practice if you spend a lot of time and effort on prototypes it is very hard to convince executive management why you are throwing them away. So my main criticism of prototypes, particularly in OO environments, is that they then tend*

to impact the design, ... because invariably the people who do the prototypes are not up with what OO is all about and have absolutely no idea of what the underlying design will be and therefore what they are asking for may be ...well not impossible ... may be exceedingly difficult to do."

An integrated development tool called ModelWorks has been used in this project. It is an active modelling animation tool which allows developers to describe the business processes and model them using the modelling tool and then animate the model. It is possible to build the skeleton of an application or a prototype as the analysis is being undertaken.

5.8.4 Project Documentation

Documentation was not available for this project *"nearly all the documentation is in fact provided by us so we actually write the business requirements document ourselves although we do it in conjunction with the users, we actually produce the document and then get them to sign it off."*

5.8.5 The Requirements Engineering Process

Elicitation

Knowledge elicitation or requirements gathering is explicitly undertaken. There is usually a gap analysis providing a list of the major features that need to be built for the client. Following on from there is a business requirements document. In association with that there is a business rule document which says for each requirement what the rules for that requirement are and that is very explicit.

Knowledge elicitation is undertaken by conducting interviews with the users and is iterative. A normal "gap" analysis would probably take 2 to 4 weeks. The

users are interviewed several times to clarify points though not for a gap because it is high level. To create a business requirements document requires 2 to 4 weeks of work.

"The requirements are done to quite a detailed level. There are certainly iterations. We go back. It can be in a number of forms. What we may do is in fact have a larger group of users where we get a large quantity of information. Then a lot of the interaction may be with only one or two [users] with again follow up presentations to a larger audience. ...we like to identify key users. What often happens is ...if we are doing a lot of development rather than a straight implementation we would encourage the users to set up within their IT group if they have an IT group or within their business unit if they don't. So they would be charged with perhaps sometimes collecting 8 or 10 various views from within the company consolidating and working with us."

There is a group of users who is the development team's interface into the company. *"What it does is that it tends to give them control within their environment in terms of co-ordinating views so that we don't go, and hopefully we don't go, into a room where there are 15 different views. Some of that has already been resolved and what we're doing is just a fine tuning of slightly different requirements which we know users ... that they are not completely opposed in their viewpoint."*

Elicitation is seen by this consultant as explicitly object-oriented *"When I do requirements documents I do [think object-oriented] if it's requirements [gathered about] something that already exists. If it's something new again I probably do, I start to think about what they are really after and then how to group things together."* The consultant starts to identify high level abstract objects at the elicitation stage *"I certainly would but the majority of the business analysts and some of the other more IT oriented people probably not. The aim*

of the document is not really to be object-oriented in any way. It is simply gathering what the requirements are."

The following exchange illustrates the consultant's response to the notion of mental modelling ***"A couple of other consultants have said that they actually start building mental models at this [knowledge elicitation] stage."***..."Correct." ... ***"That they don't put anything on paper – that these mental models are sort of living in the back of their mind and as new information comes in that it sort of alters or adds to that mental model they have"*** ... *"I do"* ... ***" and its very abstract and not really anything that they would put on paper or show anyone. Do you do that?"*** ... *"I do that. ...That's not true of some of the others. You can tell when you look at their work that they've not actually thought about any form of underlying structure at all. All they've really done is try to gather the business requirements."*

Modelling

It is not until the requirements gathering is finished that the consultant or team starts modelling *"....all through requirements gathering we are talking textual, primarily."* There are several models developed depending on the project. *"We are using UML and we don't use a lot of the diagrams and we are slowly trying to use more but at the moment the pressure is to deliver, not to model. Modelling takes time and that's difficult on highly commercial projects. Invariably the delivery date precludes detailed modelling. We do class diagrams. We are trying to do more interaction diagrams, or collaboration diagrams and we do at the very lowest level, we resolve an entity model."*

There are static models and dynamic models. And the static or class diagrams models tend to come first. *"... the static model, the object model, class diagram, however you want to describe it, is the core. Everything starts there. That is the original component with these models. And then from that springs*

collaboration diagrams and interaction diagrams, if you want to go through and do state change diagrams etc."

Modelling is based on the UML notation not the complete Rational development method. *"All of the diagramming is UML. UML is a notation; it's not a development methodology. Rational will sell you, I forget what their other tool is but, which is a development methodology and my understanding is that Jacobson is currently putting together a new one which is all based around use cases. So it is a formal development life cycle methodology about how to go through the whole process using UML as the documentation method."*

The consultant has not and does not use a complete proprietary or commercial methodology because he believes that they are too expensive and too complex *"But in most areas ... I've not seen anybody use the big methodologies. I think there are two reasons. (A) If you buy the professional ones they charge too much, which is also why I think why everyone talks about Rational, although we tend to use a competitive product, Select, mainly because it's a little cheaper. And even then I don't have as many copies as I should have because it's so expensive. They are VERY high cost and if you put on the process flow modelling, all the methodology on top of that ... What happens is that the cost of setting up a developer starts to become prohibitive and there's no return on that so either you escalate the price of your product to cover that high cost or you hope you work for a multi billion dollar company that can afford to simply write cheques and say 'yeah we will spend all this money'. The other reason certainly why we tend to use our own methodologies which have short cuts and work arounds and all sorts of different things and why even methodologies where you are supposed to follow them [in our case] there are odd documents missing and some are much shorter than they should be. There is simply not enough time to follow the whole box and dice and produce all of the documents. You produce those documents where, if you've got to get a user*

to sign off do those. Why, because that affects the bottom line and that's really what it's about."

When asked if he or the team ever showed UML type models to the users he said that it depended on the users and in his case he has only shown models to users who are familiar with the notation *"No. We would probably not show ... I don't believe we would show, a user any of the modelling that we would do as part of a top level design or a detailed design. We may show them some very high level diagrams but that's more to get operations people a view of what processing is about. ... The end user is invariably, if the data is not on a GUI somewhere they probably wouldn't have a clue that we were storing it."*

In this case the class models and interaction models were never shown to the users because they were not designed as end user models but are designed for the development team, and were passed on to the design and implementation phase.

This also led to a discussion on how difficult it is for some people, users as well as professional developers, to think in object-oriented terms *"The biggest, if you like, misnomer that I've found with OO techniques is they're not faster, they are slower. I've worked with COBOL systems, RDBMS based systems, OO systems using a very advanced IEM artificial intelligence based 4GL and then C++ and COBOL and the 4GL, an AI based one which is an object...based around objects, i.e. OO, are the fastest in development but for some simple things, some simple solution you could build it an awful lot faster out of COBOL than you could out of C++ and with a much lower, I won't say level of developer, but your developers don't need to be so bright and smart using some of the other tools." ... "**They can be trained up to produce a solution with ...** " ... "Correct. You can get people who are good solid developers, who are very productive and will do an excellent job much more easily out of some of those tools. I have found particularly with the main, or what's now regarded as the main OO*

language, being the dreaded C++ is that it is difficult to learn. It requires particularly bright people in order to do things. They tend to wander. They tend to have their own agenda about what they want to do and how they want things to work. You can't be...I always consider myself as more of a purist and try and think OO rather than think C++ and they don't, they think C++ ..." ... ***"the mentality of the old hacker programmer"*** *... "Absolutely. And it's very much like that, whereas I thought we were getting away from that and we were trying to have, from my perspective, language independent software solutions are where we should be going and we are not doing that. We are not getting to language independent because everybody now likes C++. UML was originally based on C++ implementations so it forces you down this programmer, hacker mentality not thinking about business objects or how they interrelate and how they react and actually build them from that perspective and then who cares what the language is." ...* ***"It is so easy to just set up a prototype of the screen, stick a few components on it and a bit of code behind it and that is not the way to build systems although I think a lot of systems are being built that way because ..."*** *... " I would suggest that probably 80+% of systems at the moment are built exactly in that manner."*

Use cases are not used much by this consultant except at the validation stage *"We certainly do some when we are walking through, perhaps later once we get down and perhaps have got a prototype and some rules and requirements the users now start saying well, what if ... We verbalise or walk through a scenario and then say well how does this system handle this. They may write it in an email but it's not a formal scenario driven approach."* Further questioning on the use of use cases led to the following exchange

"... I like them and I wish we could use them more often. There are not a lot of tools that support them, if you want to interact at the tool level. They can work if you document them and effectively they become a test scenario and they are certainly very good for that. And I certainly would try and get users to come up

with a specific case to actually document and if you have a formal test environment (they do) or test cases all documented and all of that. Using use cases to help elaborate the solution is often a little more difficult. Again it relies very heavily on having, using the full raft of tool sets and again I would say that most people don't use the full raft of the toolsets. They use the toolsets for as much or as little as you need in order to build the solution. At the end of the day, the built solution is what is important and enough documentation to support that."

"I had one consultant say that he thought that they were very useful and necessary but they could be dangerous when depended on in isolation because they seem to be becoming popular in some of the smaller consultancies to use them alone (as the model) because the users can understand it."

"We certainly wouldn't ...we certainly use them to highlight specific cases that the user has ...a scenario with the user has come up where they say this is really some complex or unusual process. I tried to use them for some more simple ones but they become, I think, then too simple. I actually quite like the OOIE event diagrams because effectively you are process modelling. And users can understand process models much, much better than ... I mean you show them a class diagram with all it's interactions and they go 'What does it mean?' because they can't navigate through that whereas with an event diagram in the OOIE which is more of a process model they can follow it through. They can see we are doing this operation. Which object does this operation live on? They don't care and therefore you can conceal that from them and all they are interested in is 'When we do this, this causes this to happen' and they can follow process flow through. What interactions are occurring underneath..."

Active questioning about how equirements are communicated back to the users/clients led to the following exchange:

"I'm interested in how professional people communicate their specification or their models back to the users and say 'this is what we are building for you'. And in some cases it's use cases ...and some people use ad hoc diagrams and pictures and some people just use textual specifications. So I'm just interested in that communication back to the client of 'what we are building'".

"Initially what the users are expecting is really based on the business rules, the requirements and the prototype if there is a prototype. We do a large part of the sessions with the prototype with the users, the screen prototype, and the users get used to that which is the other reason why you can't throw it away. Because at the end of the day they know what it looks like and therefore when you go in to testing that is what they expect to see. In fact they complain if there is even the slightest change. They actually ..."

"Yes. .'This is a different colour to last time'..."

"Absolutely. Recently one of the guys in Hong Kong ...I did the prototype and the Business Requirements and Business Rules, set the whole thing up to be built and they moved two non-editable fields and the users when they went up to Hong Kong to have a sort of quick check said 'where are those two fields?' (Laughs) 'Oh we put them on a different tab'...'But why?' So they are very much keyed into if you work with them heavily on the prototype that's what they expect to see as the final design regardless of what anybody maintains or says 'Oh no, no, no it can be different' at the end of the day it's got to not be. Again that's probably... back to your other question as to why it is perhaps helpful if I get involved and do them because I do think a little bit about the objects. I tend to get the GUIs reflecting perhaps a little bit more as to what the underlying structure will be and therefore it is less prone to have ...for someone to say 'It's just too hard. we need to build it a different way'."

Instead of object-oriented models various diagrams were shown to users

"We would walk through all of those diagrams with the users and in fact you can store canned scenarios (using Modelworks) that will show the navigation through and we actually will with users come up with complex cases and business cases ... You would actually walk through the end solution because the event diagram is what is built into the solution so they can very much see what they've got. Using UML the requirements and the business rules. (A) We don't have a business process model like that so there is really nothing you can show the user. Use cases would be good but again it relies on having all of the other models done because you have so many models. Again they tend to get left out. I'd like to see them used more."

The consultant believes that as a professional much of his requirements technique comes from knowledge and experience on other projects He believes that it is a cumulative thing where having seen something or solved a problem before he can solve another similar problem. *"It works in two ways. There is using your experience to recognise 'Hang on I've seen something like that before ...Yes, I recognise what it's doing therefore I can do it'. Secondly I know ... I don't have the same view of the business as a user because I'm used to the view from the software development side. Most of the users I talk to have not been in it for eight and a half years so I actually have a fairly good understanding of the development and SE or IT or whatever you want to call it, that the software development side of the business I understand and therefore when I'm requirements gathering I actually understand what the user is talking about and can actually relate that to software development so therefore I can make sure I try and work my requirements gathering around 'OK, if I had to build that what information would I need? Have they given me enough? If I built that what are the exception cases? Are there this, this this...' There are some of the questions about what they want to do if it doesn't meet these criteria, all of that. And that's predominantly drawn on a nearly 20 year career and 12 years in the same environment."*

Validation

When asked, the consultant believed that it is necessary to validate the specification itself, not the just the implementation but the specification, once there are some models or the specification itself has been developed. *"Absolutely, that's exactly what we do. We asked the users to sign in blood, not that even if they sign it they will then swear black and blue that it was documented wrong. And you say 'But you signed it off, blah blah, you got it wrong. (laughs) Here's your signature' 'No, no, no ...you guys got it wrong. You told us ... how did you ever convince us that that should happen?'"*

Validation is undertaken by doing formal walkthroughs with the users *"And I've still had users sign things and then two weeks later claim that we got it wrong despite the fact that's what's in it and they said, and they agreed and signed on the specification that that was correct. And they will come back and say 'No, it's not' and at the end of the day you can't argue because you just end up with more trouble than it's worth if you highlight too heavily the fact that they had signed off."*

Walkthroughs, either with use cases or a textual based specification document is the only validation method. There is no formal validation of formal proofs performed on the specification. *" ... we go through the requirements, generally not the rules as much but mainly the requirements and probably the prototype and we would get formal sign off."*

Validation and acceptance testing are ongoing and are done at specific points during the development process *"We have ...the methodology has in it the specific points when testing is done. You build that into your test plan and you build it into your development plan as well as you know that developers are supposed to do unit testing as part of their development. At the end of the cycle you have integration or internal testing and then you have full user acceptance*

testing. We try and at some points through the development process, try and have some form of review with the users although that's not always possible and not always done. But we try just to ensure that they don't change their mind half way through, or if they do we find out about it early."

The prototype is used more in validation than use cases. The prototype is invariably used to walk through the specification with the users and use cases are used for exceptional or special cases *"The last major one I did we were looking at the requirements document but as we were looking at the requirements document we had the prototype running and projected up on a big screen and we actually walked through the prototype in relation to the requirements and when they said 'Here's a requirement that says you've got to be able to do something' you go through the prototype and say well this is how you do it."* The people involved were the same users that were interviewed during requirements gathering.

Again in this case requirements gathering is seen as ongoing throughout the system development and the consultant expects to revise or revisit the specification during the design and implementation stages. *"We know that the users will always come up with a new scenario or something that has changed or somebody forgot something. A lot of the time it's something which came up as an original requirement and everybody agreed was not important and then 3 months into development suddenly somebody thought of one particular business case which is why they had that requirement in there and that is absolutely important, it's critical. So suddenly now it is have to do it rather than be able to defer it to a later date so there is certainly a lot of that. Users get things wrong. A user will tell you that black is white or whatever. It's all perception. We certainly get a lot of that where occasionally you get the wrong user. Finding the right users I've always found to be probably the biggest trick. If you get wrong users you get the wrong answers to the right questions and then when you've built the system and you display it to perhaps a wider body*

suddenly somebody says 'Why did you do it that way, that's stupid because...' And you say 'Well that's not what we were told'. 'Oh, you couldn't have been told anything else because this is the rule that says in this piece of legislation' (laughs) or 'This is the policy manual that says that we do it some other way'."

The consultant also does not believe that automated tools solve these problems *"I can't see how ANY tool can ever protect you from a user giving you or telling you how something is ...what the rules for something are or the rules for a particular interaction is and simply telling you the wrong thing and not knowing that they are telling you the wrong thing. And then at the end of the day your system does the wrong thing. It does EXACTLY what you were told but its wrong and another user as soon as they see it will tell you that it is wrong. And it is simply impossible to protect from that and you always get that. I can't see how, using a tool, how you can eliminate unfactual information."*

5.9 Summary

This chapter has described a multiple sequential-case study of modelling for object-oriented requirements engineering. This case study was undertaken to answer the research questions posed in Chapter 3 and investigate the validity of the concepts in the conceptual process model proposed in Chapter 4. The data collected in the six case studies has been organised into six case study descriptions which were presented in this chapter. Cross-case analysis of the case study data appears in the next chapter.

Each of the cases involved the investigation of the use of models in the requirements engineering process within the context of a specific project, the methodology used and the approach of the analyst or consultant. The case study descriptions show that in Case 2 it emerged that use case models were the main vehicle for describing and communicating the requirements to users. This use of use case models was not explicit in the initial conceptual model and was

followed up in subsequent cases. Case 3 revealed a comprehensive description of the development of mental models during the elicitation phase and this concept was explored in subsequent case studies. Case 4 revealed extensive use of rich pictures, adhoc diagrams and screen simulation prototypes for describing and communicating the requirements with users and this was also followed up. Case 5 and Case 6 provided more rich detail and also reinforced some of these concepts contained in or emerging from the conceptual process model (such as mental modelling, prototyping and the use of informal models to help users understand and discuss the requirements) but did not reveal new areas for exploration.

The next chapter analyses the case study data in terms of addressing the research questions, investigating the empirical validity of the initial conceptual process model and its revisions.

Chapter 6

Case Study Analysis

6.1 Overview

Chapter 6 presents the qualitative analysis of the sequential-case study data in terms of three objectives. The first objective is to determine the empirical validity of the initial conceptual process model presented in Chapter 4 of this thesis. This analysis is described in section 6.2. The second objective is to analyse the emerging characteristics of the evolving versions of the conceptual process model that are revealed and/or reinforced by the sequential-case study. Each new version of the conceptual process model and the case study data that produced it are described in section 6.3. The third objective is to describe and discuss findings from the sequential-case study data that relate directly to the research questions which explore the opinions, beliefs and behaviours of the professional analysts engaged in object-oriented requirements engineering. The analysis of these findings is described in section 6.4. The overall findings are summarised in section 6.5.

The conceptual process model presented in Chapter 4 of this thesis was based on the literature regarding object-oriented requirements engineering and

together with the research questions presented in Chapter 3 of this thesis provided the basis for the interview scripts used in the sequential-case study described in Chapter 5 of this thesis. The categories used in the sequential-case study represent the technical, cognitive and social concepts embodied in the conceptual process model and the research questions. The concepts in the conceptual process model emphasise the technical aspects of the object-oriented requirements engineering process and the concepts in the research questions emphasise the cognitive and social aspects including the opinions, beliefs and behaviours of the professional analysts who undertake object-oriented requirements engineering. Both the conceptual process model and the research questions address aspects of the three processes of object-oriented requirements engineering: elicitation, modelling and validation.

Qualitative data analysis methods based on Miles and Huberman (Miles and Huberman, 1994), Fitzgerald (1997) and Wynekoop and Russo (1997) have been used to organise and classify the sequential-case study data. The analysis presents the researcher's understanding and interpretation of other peoples interpretations and perceptions as expressed in a number of iterations and interviews (Walsham, 1995). Both within-case analysis and cross-case analysis (Cavaye, 1996, Miles and Huberman, 1994) have been used to highlight reinforcement of existing conceptual categories and to identify emerging conceptual categories.

The two main documents used in the data analysis are the interview script and the categorisation document. The initial interview script (see Appendix A.1) was partitioned into seed categories based on the initial conceptual process model as outlined in Chapter 4 and the research questions as proposed in Chapter 3. The initial categorisation document contained the same categories as the interview script. For each case the transcript data was placed into the categorisation document under the appropriate category headings. The process of transcribing the data and placing the transcript data into the categorisation

document revealed new categories or sub-categories. These new categories or sub-categories were then incorporated into the subsequent interview script so that as the sequential-case studies progressed the interview script grew to incorporate emerging categories and corresponding questions for the participant. The final interview script (see Appendix A.2) shows the extra categories and corresponding questions that emerged during the six sequential-case studies.

6.2 Validation of the Initial Conceptual Process Model

As stated above the first objective of the qualitative analysis is to empirically validate the initial conceptual process model (see Figure 6.1) presented in Chapter 4 of this thesis. The initial categories identified from the initial conceptual model and associated interview questions are given in Appendix A.1. The main categories addressed in the initial conceptual process model were:

- The use of three pocesses: elicitation, modelling and validation
- The use of feedback in elicitation for clarification
- The identification and use of explicit static and dynamic models

These categories were addressed in all six cases. The following three sections present the analysis of each of these categories and the findings are summarised in Table 6.1

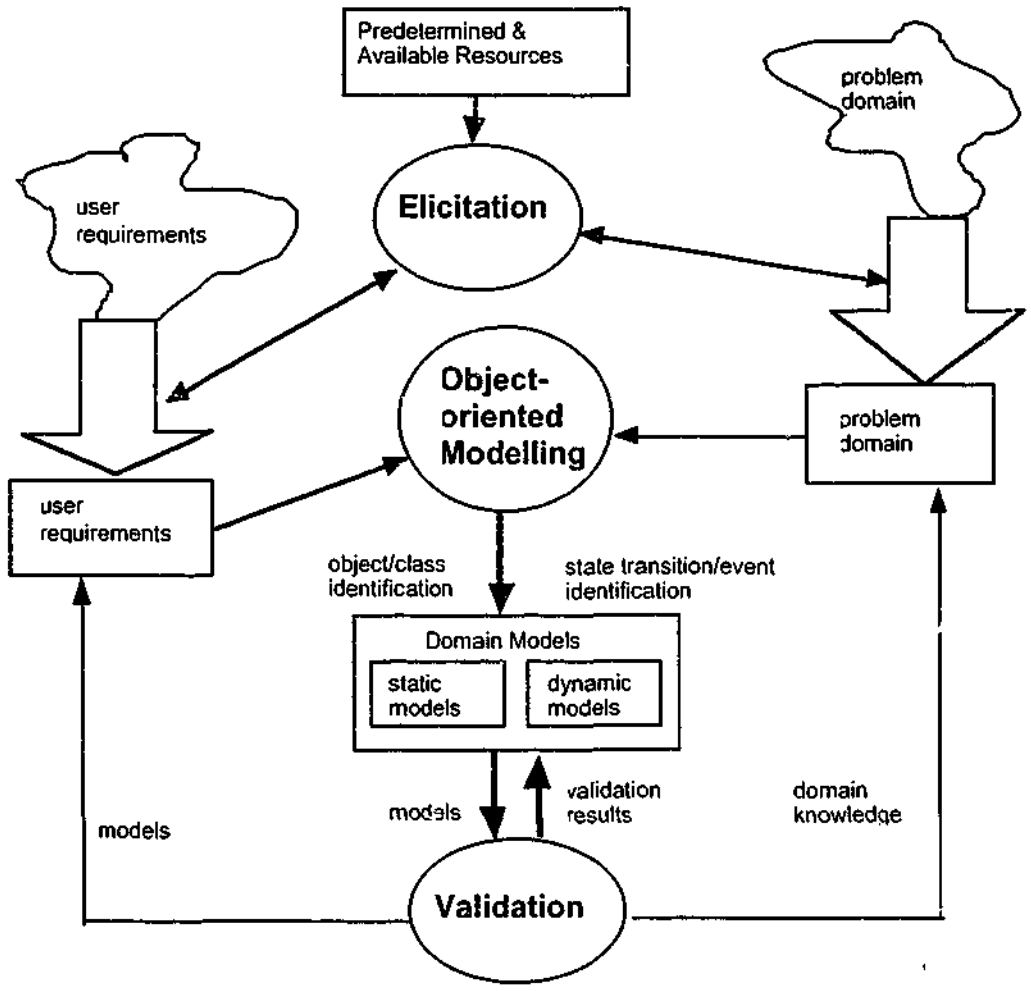


Figure 6.1 Initial Conceptual Process Model

6.2.1 The use of three processes: elicitation, modelling and validation

One of the major concepts embodied in the conceptual process model is that there are three identifiable processes in the object-oriented requirements engineering process. Those processes are elicitation, object-oriented modelling and validation of the models. The specific questions addressing this concept taken from the interview scripts are:

- Is elicitation explicitly undertaken and when does it start?
- When does modelling begin? That is, when do you start drawing object models?

- Do you think it is necessary to validate the specification once the models have been produced?
- When does the validation process start?

The three processes of elicitation, modelling and validation were identified in all six cases. In Case 1 there were three stages within the elicitation process. Elicitation started with a "blast-off" meeting and continued with regular interviews with the users and meetings of the project team. This was followed by an apprenticing activity and the specification of individual requirements using requirements cards as described in section 5.3.3. There was little traditional modelling although cards were used to represent requirements and their characteristics. Validation was based on walkthrough techniques.

In Case 2 elicitation was initiated by interviews with upper management to *"scope the project"*. Existing documentation and data models were used as a starting point for interviews and setting up use case scenarios. Modelling was based on standard OMT models and use cases. Validation was based on walkthrough techniques.

In Case 3 elicitation was done by interview and prototyping with a small group of selected users and a specific *"subject matter expert"* as described in section 5.5.5. Modelling was based on OMT object/class models and comprehensive textual use cases. Modelling and prototyping were supplemented with a case tool called Software Through Pictures (STP). Validation was based on walkthroughs and revisiting the use cases and the prototype with the user group and subject matter expert.

In Case 4 elicitation was done explicitly where possible but because this project was a generic package the users were not captive to the organisation. For the purposes of elicitation, in-house business analysts and pre-sales people played the role of users in a client organisation. Both entity-relationship modelling

and UML models were used to produce requirements models. The consultant also used some use cases together with ad hoc diagrams and rich pictures as models with the users. Validation was done as walkthroughs with the prototype and role-playing based on use cases.

In Case 5 elicitation was done by interviews with an emphasis on building trust and rapport. There was no use of specific modelling techniques or notation. The specification was based on a document which embodied concepts that could be directly implemented by experienced system developers. This appeared to be based on a "programmer-oriented" approach to system development. Specific validation was not seen as necessary if the team has *"built the right thing"* although some acceptance testing was seen as useful.

In Case 6 elicitation was explicitly undertaken using interviews and involved gap analysis and the production of a business requirements document. Modelling was based on entity-relationship models and UML notation though not a lot of models were produced. Use cases were not used extensively because there are not a lot of tools to support them. Use cases were seen as useful in the validation process where the team can *"verbalise or walk through a scenario"*. Prototype demonstrations were also considered useful for validation.

The analysis of findings for each case are summarised in Table 6.1. They show that although the methods used in each process in each case were different, the three processes could be identified in all cases and so the existence of three processes was empirically validated.

Case No	Elicitation	Modelling	Validation
Case 1	Blast off meeting Apprenticing Requirements cards	Flow charts	Walkthrough
Case 2	Interviews and graphical use cases	OMT models and use cases	Walkthrough
Case 3	User group Subject matter expert Interviews Prototype	OMT models Textual use cases Case tool Prototype	Revisiting use cases and prototype Walkthroughs
Case 4	Role playing by in-house business analysts	ER and UML models Ad hoc diagrams and pictures Sparing use of use cases Prototype	Walkthroughs based on prototype and use cases
Case 5	Interviews with a business focus	No specific models Text-based specification	No formal validation Some acceptance testing
Case 6	Interviews Gap analysis Business requirements document	ER and UML models Some use cases Prototype	Walkthroughs Some use cases Prototype demonstration

Table 6.1 The three processes of object-oriented requirements engineering

6.2.2 The explicit use of feedback in elicitation

The explicit use of feedback in the elicitation process of object-oriented requirements engineering as depicted in the initial conceptual process model as a double-headed arrow was actively explored with the following question taken from the interview scripts:

- Is knowledge elicitation iterative? That is, do you go back to the users several times?

In Case 1 there was feedback and elicitation was iterative. On-site meetings were held every second day with the clients and these meetings were interspersed with in-house project team meetings.

In Case 2 elicitation was an iterative process. The analyst would go back to the client, on average, three times so that the whole transaction specification took about a week. Information gathering usually took place on an initial half-day with some follow-up over the next two days. A week after the start the specification was given back to the client for review and the analyst walked them through it.

In Case 3 elicitation was considered to be *"highly iterative"* and included contact with the subject matter expert every couple of days.

The generic nature of the package being developed in Case 4 meant that there were no actual users available and elicitation was done using role-playing. It is not clear whether the role-playing involved iteration.

In Case 5 knowledge elicitation was seen as iterative and based on feedback. The team went back to the users several times where some piece of information triggered the need to explore some new feature or aspect.

In Case 6 users were interviewed several times to clarify points and produce the business requirements document. Different groups of users may be involved in each iteration including large groups and subsets of key users.

The analysis of findings for each case is summarised in Table 6.2. They show that in all but one case the analyst saw the elicitation process as iterative and based on feedback for acquiring and clarifying requirements. The feedback loop in elicitation as shown in the conceptual process model was empirically validated.

Case No	Feedback in elicitation
Case 1	Feedback from an iterative cycle of client and project team meetings
Case 2	Elicitation based on feedback from several meetings with clients
Case 3	Elicitation seen as highly iterative with the subject matter expert
Case 4	Difficult to say since elicitation based on role-playing by in-house business analysts
Case 5	Feedback from an iterative cycle to follow up specific points
Case 6	Iterative process with various groups of users

Table 6.2 Feedback and iteration in elicitation

6.2.3 Identification and use of static and dynamic models

A major concept in the literature regarding object-oriented modelling is that there is a need for both static and dynamic models for the representation of object-oriented concepts. The nature of, and types of, static and dynamic models produced in object-oriented requirements engineering were actively explored using the following questions taken from the interview scripts:

- Which models are produced during specification?
- Do you produce class models, use case models or interaction models?
- Would you categorise models as static or dynamic?

Case 1 did not make extensive use of object-oriented models although these models were available as part of the Volere methodology. The main vehicle for representing requirements was the requirements cards and the modelling that was done was based on flow charts. In Case 2 both static OMT object/class models and dynamic interaction models were produced although the dynamic models were put into the appendices and were rarely used with the users or for requirements specification.

In Case 3 static OMT object/class models and dynamic interaction models were developed but the interaction models were used later in the system

development process rather than in requirements specification. Case 4 included entity-relationship models and static OMT object/class models. The dynamic models included state-transition models and interaction models.

In Case 5 models were viewed as *"a set of concepts"* and these models were classed as dynamic by the analyst. In Case 6 several types of static and dynamic models were used including entity-relationship models, object models, state transition models and interaction models *" ... the static model, their object model, class diagram, however you want to describe it, is the core. Everything starts there. That is what is the original component with these models. And then from that springs forth collaboration diagrams and interaction diagrams, if you want to go through and do state change diagrams etc."*

The analysis of findings for each case is summarised in Table 6.3. They show that in four of the six cases models were produced and that they were specifically defined as static and dynamic models. The concept of separate static and dynamic modelling for object-oriented requirements engineering was empirically validated.

Case No	Static Models	Dynamic Models
Case 1	Requirements cards Flow charts	none
Case 2	OMT object/class diagrams	OMT interaction diagrams
Case 3	OMT object/class diagrams	OMT interaction diagrams (but not in requirements specification)
Case 4	ER diagrams OMT object/class diagram	OMT interaction diagrams State-transition diagrams
Case 5	None reported	Models seen as a set of concepts
Case 6	ER diagrams OMT object/class diagram	OMT interaction diagrams State-transition diagrams

Table 6.3 Static and Dynamic Models

6.2.4 Summary - validation of the initial conceptual process model

Overall, the characteristics of the initial conceptual process model were empirically validated by the results contained in the data from the six cases. There was no evidence for removing any of the major concepts from the initial conceptual model. The main characteristics analysed in this section were reinforced in five of the six cases where the five cases were different for each concept.

Table 6.4 shows a summary of the cross-case analysis of the overall findings from each of the six cases regarding the initial conceptual process model. The analysis of the findings produced several emerging concepts that were added to the initial conceptual model in three revisions of the model as discussed in the next section.

	Three processes	Feedback within the elicitation process	Explicit static models	Explicit dynamic models
Case 1	reinforced	reinforced	reinforced	
Case 2	reinforced	reinforced	reinforced	reinforced
Case 3	reinforced	reinforced	reinforced	reinforced
Case 4	reinforced		reinforced	reinforced
Case 5		reinforced		reinforced
Case 6	reinforced	reinforced	reinforced	reinforced

Table 6.4 Cross-case analysis of concepts embodied in initial conceptual process model

6.3 Evolution of the Conceptual Process Model

The following sections outline the findings of each of the six case studies with reference to the research questions and the evolution of the conceptual model. The major emergent categories which were reinforced by subsequent cases were:

- Evidence of the use of use case models as distinct models for requirements representation
- Evidence of mental modelling by analysts during elicitation before any models were committed to paper
- Evidence of the use of separate formal and informal models where informal models were the only models shown to users by analysts when discussing requirements.

6.3.1 Additional Model Type: Use Cases

Although use cases or scenarios are described as ways of modelling requirements and transactions for object-oriented systems in several methodologies, they are difficult to classify as specifically static or dynamic models. Use cases can come in textual forms, graphical forms or both. In the analysis of the characteristics of object-oriented models in Chapter 4 of this thesis, Table 4.1b, use case models were classified as static models because of their textual characteristics. They could have been classified as dynamic models based on the graphical, process-oriented representation. As a result use case models were not explicitly shown in the initial conceptual process model. The use of use case models (both textual and graphic) as requirements models distinct from the static and dynamic models emerged as a category worth exploring in Case 2. The following summary of a selection of quotes from the transcript illustrates the emergence of this category:

"The generic object model is a standard Rumbaugh et al OMT model and the rest of the methodology is built around Jacobson use cases ...The clients tend to focus on the use case components and look at the flow and the data ... In the course of the methodology development about six client organisations were used who provided real examples of transactions. These were developed into example use cases with a minimum data set that was incorporated from the

real examples.... The partial specification was presented with a use case methodology – standard pathway, exceptions, maybe an alternative for the generic case (a la Jacobson) ... The basic techniques are interviewing and working with use cases/scenarios...."

The questions incorporated in subsequent interview scripts exploring the use of use case models were:

- Do you use use case models at this stage [elicitation]? What form do they take, textual or graphical?
- Could you comment on the role and/or importance of use cases in the specification process?
- Could you comment on the relationship between use cases and static and dynamic models?

Case 1 used a methodology that potentially included use cases but the analyst did not use them in the project under study. In Case 2 the use of simplified use cases as models in requirements specification emerged as a significant technique. *"The generic object model is a standard OMT model and the rest of the methodology is built around Jacobson use cases. The IT [liaison] person communicates the information from the model to the client users. The focus is on use cases. There is a standard process modelled as a use case flow diagram and a use case [structured dialog] and the client has to tick the box if there is a good fit."*

Subsequently, Case 3 and to a lesser extent Cases 4 and 6 gave evidence of the use of use cases for requirements modelling. *"We saw the need for a static type of model and a dynamic type of model and use cases...the use case model was more stand alone and really became the functional statement that went behind or reinforced the UR [user requirements] prototype ...really most of the first*

phase of the work developed our use case model and a prototype to go with it. Those were the key vehicles for delivering the requirements." [Case 3]

"Well, that's ...according to the theory they [use cases] are the backbone – you start with them and they go right through to the testing and so forth and I think that's reasonable. Generally speaking I think it's a good way to use them." [Case 4]

"I wouldn't assert that when you build your prototype you take a particular use case and implement it at some level. I think you implement the system at some level and that would give you a capacity to traverse it using some use case." [Case 4]

"We do [use use cases] a little bit but not greatly ...We certainly do some when we have got a prototype and some rules and requirements the users now start saying well, what if ...We verbalise or walk through a scenario and then say well how does this system handle this. They may write it in an email but it's not a formal scenario driven approach." [Case 6]

Table 6.5 summarises the evidence for the use of use cases for requirements modelling. Use cases were used to model requirements in four of the six cases.

Case No.	Use of use cases as requirements models
Case 1	Didn't use use cases
Case 2	Simplified use case diagrams and simplified use case dialogues in an in-house template requirements specification methodology
Case 3	Use case scripts together with a prototype
Case 4	Use cases occasionally used together with a prototype and ad hoc diagrams
Case 5	Didn't use use cases
Case 6	Used use cases for dealing with exceptions or in special cases

Table 6.5 Evidence of use cases for requirements modelling

This provided sufficient reinforcement to include use cases as a separate type of model in the conceptual process model. Figure 6.2 shows version 2 of the conceptual process model which incorporates use case modelling explicitly.

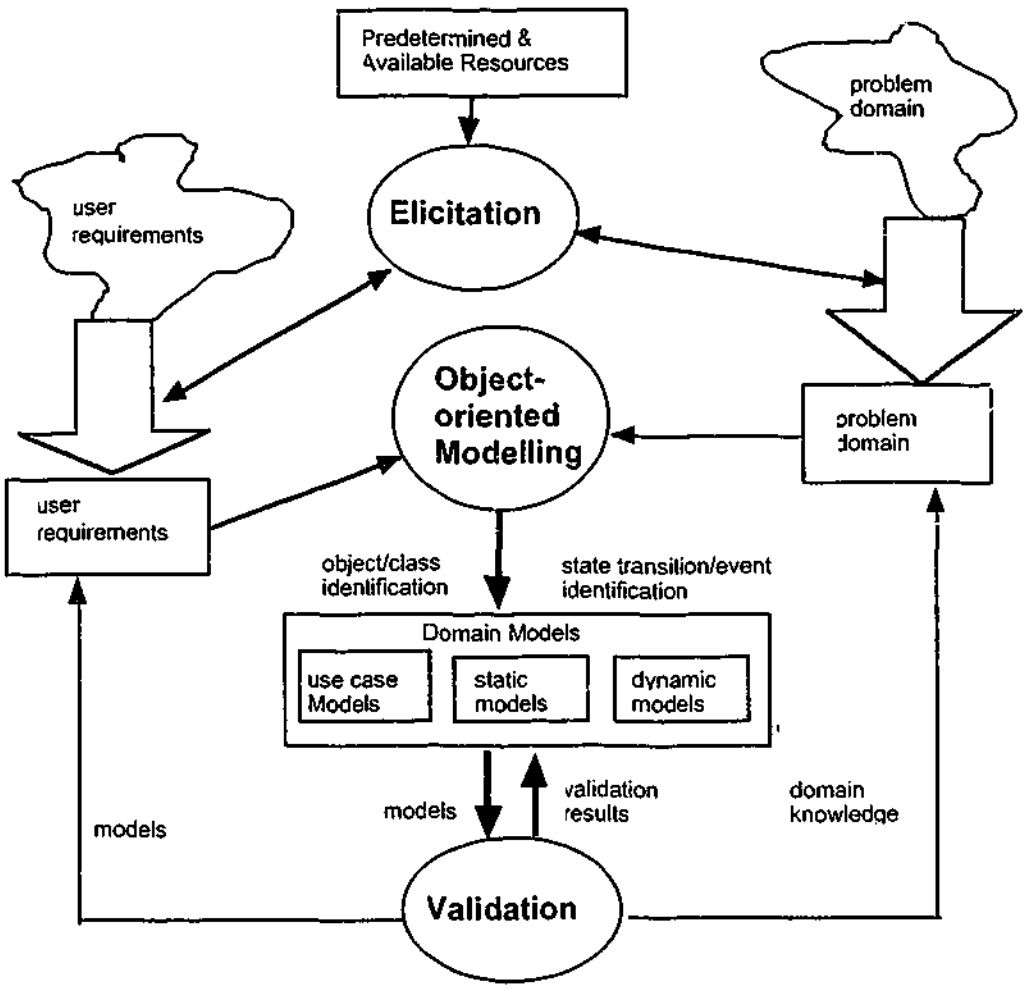


Figure 6.2 Version 2 of conceptual process model

6.3.2 Mental Modelling

Mental modelling in the process of requirements elicitation and specification was previously addressed in section 2.3.3 in Chapter 2. The concept of mental modelling during elicitation by an analyst emerged in Case 3 as the following selection of quotes from the transcript illustrates: *"...there were fragments of that [business] model getting developed in a couple of people's heads for*

probably three months...I would say that it was being done largely privately and it was not written down until the last minute when it was just a dump"

"You will be listening very carefully [in project meetings] and collecting and cataloguing constraints and refining the abstractions in your mind."

This concept was explored in subsequent case study interviews with the following question:

- Do you start developing mental models during elicitation?

Overall four of the six analysts believed that they were continually "modelling in the mind" during the elicitation process and that these mental models were further refined in the mind before they were communicated to others (users or fellow analysis team members) or before they were committed to paper.

In Case 4 creating mental models involving key objects was seen as a natural part of requirements elicitation and modelling *"I think that I do immediately start thinking of key objects during requirements gathering, not in any formal way, they just pop into one's head. I don't agree with the implication ...that identifying objects and 'building mental models of the system' are mutually exclusive. One can help the other."*

In Case 5 mental modelling was perceived as an integral part of abstraction *"This is really about how people think. Some of us more conscious of the models, others not."*

In Case 6 producing mental models during elicitation was seen as a natural way of thinking for that particular analyst although he thought it was a personal thing and that other analysts might not work that way *"I do that. ...That's not true of some of the others. You can tell when you look at their work that*

they've not actually thought about any form of underlying structure at all. All they've really done is try to gather the business requirements."

Mental modelling emerged as significant in Case 3 and was subsequently reinforced in Cases 4, 5 and 6. Table 6.6 summarises the findings regarding mental modelling.

Case No.	Use of mental modelling
Case 1	Didn't mention mental modelling
Case 2	Didn't mention mental modelling
Case 3	Mental modelling emerged as significant
Case 4	Mental modelling seen as a natural part of requirements modelling
Case 5	Mental modelling perceived specifically in terms of abstraction
Case 6	Mental modelling seen as natural to analyst personally (not necessarily for other professional analysts)

Table 6.6 Evidence of mental modelling

There was sufficient reinforcement to incorporate mental modelling explicitly in the conceptual process model. Figure 6.3 shows version 3 of the conceptual process model incorporating mental modelling.

6.3.3 "Formal" and "informal" models

As described above, when asked when the modelling process began, most analysts said they were building models in their heads long before any formal models were written down. "Formal" in this context is not used in the same way as in mathematical or computer science literature where it means a mathematically based model. It is also not used in the same way as Pohl (1994) or Jarke (1993) use the term where it refers to formal specification languages such as Z (Spivey, 1989) and VDM (Bjorner and Jones, 1987)`. Rather, it is used to indicate models which use specific notation eg. OMT or UML diagrams.

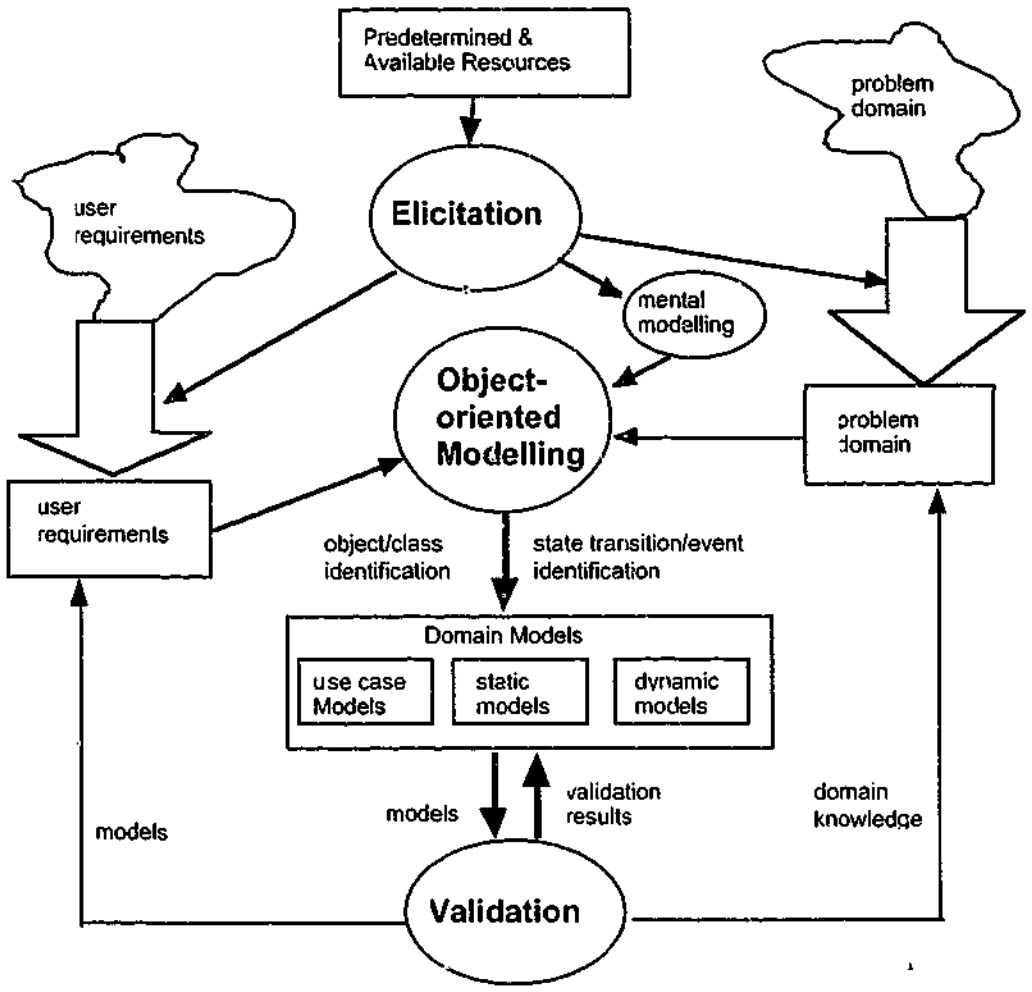


Figure 6.3 Version 3 of the conceptual process model

Evidence of the use of informal models (simple use case models, adhoc pictures, diagrams, animations etc) instead of formal notated models (OMT/UML object /class diagrams) for communicating the specification to clients and users emerged in Case 4. Subsequent reflection and re-examination of the transcripts of Cases 1, 2 and 3 revealed further evidence of this "separation of models". For the purpose of discussion formal models and informal models are distinguished in the following way. *Formal* models are considered to be those models that require training in order to be understood or explained. That is, models that contain specific, often graphical notations such as OMT models, UML models, interaction models or state models. *Informal* models are considered to be models that can be understood and explained without specific training. In this category are natural language models

including text descriptions, use case scripts, ad hoc diagrams and interactive demonstration models as often produced for prototypes.

Questions regarding types of models which were part of the original interview script were:

- Which (how many) models are produced during specification?
- Who uses them?
- Who are they produced for?
- Which models, if any, are shown to the user?
- Which models are used internally by the development team?

Questions regarding types of models asked in interviews subsequent to Case 4 were:

- Which (how many) models are produced during specification? Do you produce class models, use case models or interaction models?
- Do you use informal models (pictures etc) to communicate with the users?
- Do you use use case models at this stage? What form do they take?

Evidence of the use of separate models is shown in Table 6.7.

Case No.	Models shown to users	Models used in design and implementation (not shown to users)
Case 1	Requirements cards that were part of the methodology	Unknown
Case 2	Simplified use case diagrams & dialogues	OMT class & interaction models
Case 3	Use case scripts & prototype	OMT class & interaction models
Case 4	Ad hoc diagrams, a prototype (in PowerPoint) and some use cases	OMT models
Case 5	Text document only	No models – the system was implemented directly from text document
Case 6	Simplified Odell event diagrams, animations, prototype and use cases for exceptions & special cases	OMT class & interaction models

Table 6.7 Evidence of the separation of models

Three of the cases (Cases 3, 4 and 6) used prototypes early in the specification process. In another case (Case 6) an animation package was used. Another case (Case 4) used a standard presentation package (Power Point) to produce sample screens that were worked through with the users. All but one of the object-oriented analysts produced use case scripts and one also used use case diagrams for informal modelling with clients. Some (Cases 1 and 4) also used ad hoc diagrams and one (Case 6) used rich pictures (Checkland and Scholes, 1990) to explain and communicate the specification to users.

In the case studies there seemed to be two different kinds of modelling taking place. Firstly, there were the informal models that were used to communicate with the users. Secondly, more formal models were developed which were not shown to the users because it was believed that the users would not understand them. The formal models were developed primarily for design purposes and were private to the analyst or team of analysts. In effect these formal models were the analysts' internal version of informal models.

The mapping process of informal models to formal models appeared to be based on using the informal models to clarify requirements with the users and then to refine the formal models accordingly. When asked how this mapping process was carried out most analysts had difficulty describing the process. It was described as being able to think in terms of either type of model depending on whom the analyst was talking to at the time. It was also described as "an automatic translation process", going from one model or group of models to the other as appropriate.

In Case 1 requirements cards were used directly with the users to represent or "model" requirements and it was not made clear which models, if any were used in the design and implementation process *"The card is pretty much self documenting ...straight into the actual requirement spec. So once you have the cards complete, a lot of the hard work is done ... we cannot write the requirements, they must tell us the requirements ... we work with them on the cards. No point in us trying to tell them what they need. They need to tell us to allow us to document it."*

In Case 2 only simplified use case diagram and dialogues were shown to the users when describing requirements. Models based on formal notation (OMT) were considered too complex for users to understand *"We tell them [the users] that the model is technical mumbo jumbo."* ... ***Is that in general your experience that object type models can't really be shown to users*** ... *Well you know I wouldn't show them a data model either ...the closest I've gotten is working with this type of flow diagram (use case flow diagram)...they can follow that pretty well but they don't usually have the patience to really work through the interaction diagrams or the model. It just takes too much explanation."*

In Case 3 use case scripts and a prototype were used to develop the requirements with users and formal OMT models were not shown to users

Would you, at any point, have shown this group [the user group] ... object models or use case? *Yes.* ***Did they understand how use cases worked and so on?*** *Yes.* ***And what about the OMT model or the interaction model?*** *We would have stopped short there.* ***Is this because you do not think that the users/clients would be able to understand the OMT models (say without extensive explanation or training),*** *... Yes. Unless the 'users' were IT-literate people, which most aren't.* ***... or is it because you have tried showing these types of models to users/clients in the past and they haven't understood them?*** *I don't think I have ever tried, at least not with real business users. I presented an Object Modelling Workshop for several years, and I can assure you that it takes a surprising number of IT people several days to understand the basics of conceptual modelling (class versus instance, relationships).* ***Do you believe it is not necessary to show them?*** *I believe it is not only not necessary, but potentially dangerous. It is the analyst's job to perform the use case to business object model translation.*

In Case 4 the analyst was explicit about using various ad hoc diagrams, pictures, PowerPoint simulations and some use cases to communicate the requirements to users *"I mean if you draw a picture and that doesn't make any sense to them then you draw another one ... A requirements specification has to be in terms that they understand and those three mechanisms we've already mentioned are the way: the use case, the ad hoc diagrams and the dynamic screen simulations... and the encompassing text too, you don't write your requirements document in use case speak from start to finish."*

"I have tried showing these models to users and it hasn't worked well because many people find such abstractions hard to relate to. My former colleagues at [X] found the same thing with data models. One consultant... tried replacing the entity boxes with evocative pictures and achieved better communication.

I am not saying one should do away with the formalisms. They are a powerful aid to one's own understanding and analysis but they are not a good tool for feeding requirements back to the users. It's much better to bend the formalism to the user than the user to the formalism."

The consultant in Case 5 did not use models or explicit diagrams. Most of the specification was based on text-only documents.

In Case 6 the most important objective was to get the users or clients to sign off on the specification *"...we would have a formal walkthrough with the users where we go through the requirements, generally not the rules as much, but mainly the requirements and probably the prototype and we would get formal sign off"*. In this case the prototype was the most used tool for validation and use cases were only used for exceptions or special cases *"...as we were looking at the requirements document we had the prototype running and projected up on a big screen and we actually walked through the prototype in relation to the requirements"*.

Figure 6.4 shows version 4 of the conceptual process model showing the separation of formal and informal models in the object-oriented requirements engineering process. The feedback from the validation process for formal modelling in the problem domain is via the ad hoc mapping of informal models to formal models for use in design and implementation.

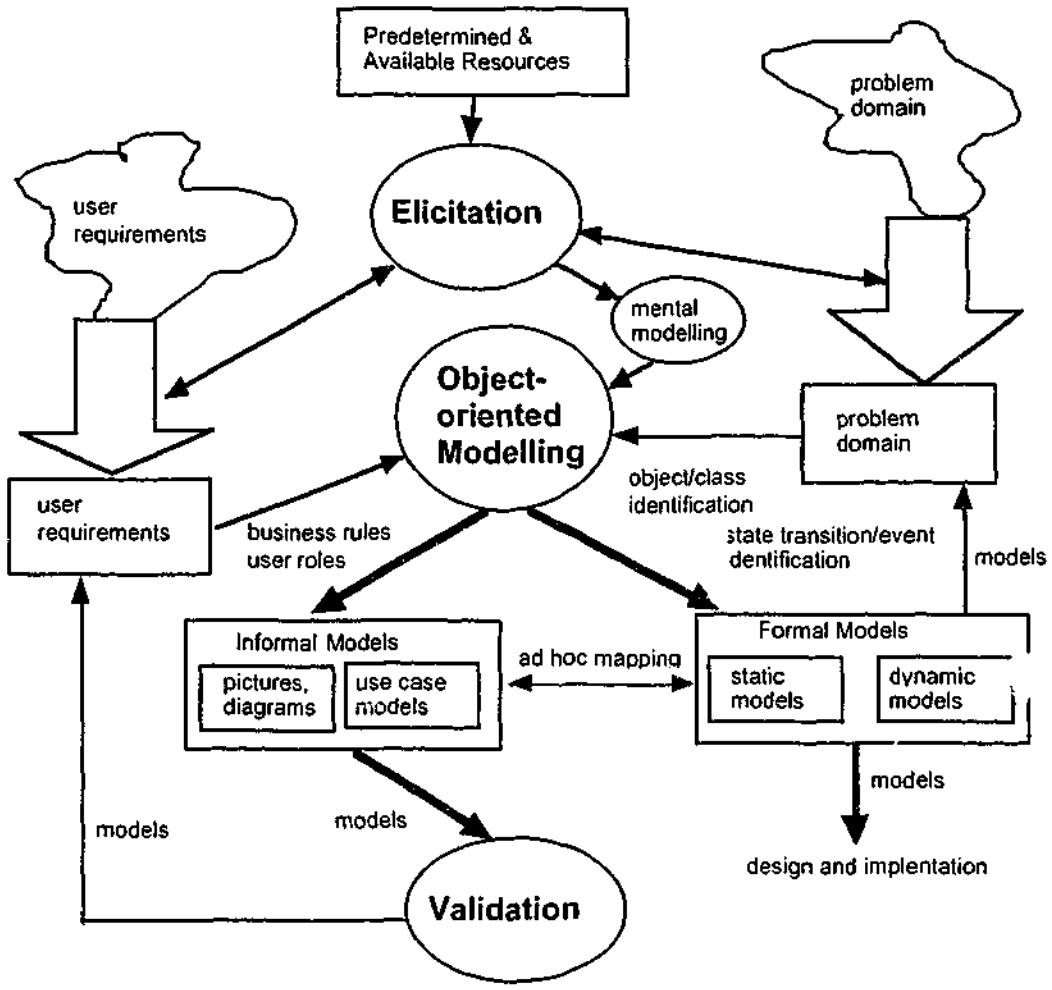


Figure 6.4 Version 4 of Conceptual Process Model

6.3.4 Summary - evolution of the conceptual model

This section summarises the evolution of the conceptual process model case by case. Case 1 provided reinforcement of the initial conceptual model. Case 2 revealed the use of use cases as an informal modelling method. Case 3 reinforced the use of use cases as informal models and revealed the use of mental modelling by the analyst before any formal modelling began. This led to version 2 of the conceptual process model. Case 4 reinforced the use of use cases and mental modelling and revealed the use of rich pictures for informal modelling which reinforced version 2 and led to version 3 of the conceptual process model. Cases 5 and 6 reinforced the notions of mental modelling, the use of informal models based on use cases and diagrams and pictures for

communicating the requirements to the users and the use of formal models based on the standard notations of ER, OMT and UML for modelling the requirements for design and implementation. The last two cases also did not reveal new avenues for exploration in relation to the conceptual process model or the research questions although there were some other interesting areas worthy of investigation outside the boundaries of this project. These further potential areas of research are discussed in Chapter 8.

Table 6.8 Summarises the evolution of the conceptual process model from emerging categories that were incorporated into subsequent interview scripts. Reflection and re-examination of the transcript data revealed reinforcement of the use of separate models for communicating the specification to users and design and implementation professionals.

	use case models	Version 2 of conceptual process model	mental models	Version 3 of conceptual process model	formal and informal models	Version 4 of conceptual process model
Case 1					informal models for users reinforced by reflection and re-examination	
Case 2	**revealed**				reinforced by reflection and re-examination	
Case 3	reinforced		**revealed**		reinforced by reflection and re-examination	
Case 4	weakly reinforced		reinforced		**revealed**	
Case 5	reinforced		reinforced		weakly reinforced	
Case 6	weakly reinforced		reinforced		reinforced	

Table 6.8 Emerging concepts from sequential-case studies as incorporated in the conceptual process model (Figure 6.4)

6.4 Findings related to the cognitive and social aspects of the requirements engineering process

Although the research questions proposed in Chapter 3 were developed to aid in the empirical validation of the conceptual process model as discussed in sections 6.2 and 6.3 above, the research questions were also designed to explore the opinions, beliefs and behaviours of the professional analysts engaged in object-oriented requirements engineering. The following sections describe and discuss the cognitive and social concepts that arose in the sequential-case studies which were independent of the conceptual process model. The main categories were:

- The analyst always thinking in object-oriented terms
- The use of in-house methodologies
- Evidence of opportunistic approaches to elicitation

6.4.1 Analyst always thinking in object-oriented terms

One concept specifically explored by the research questions was the notion that object-oriented system development involves a new way of thinking about system development (Budd, 1997). The question asked in interviews was:

- Is elicitation specifically object-oriented, i.e. do you think "OO" from the start?

The analyst in Case 1 did not think in object-oriented terms at all. In Case 2 elicitation in general was not seen as specifically object-oriented but the analyst did see elicitation as object-oriented for her personally because *"...that's the way I think ... so it's hard for me to unbundle it... we don't say to them we are really*

talking about objects and we are using an OO methodology. We just do it and they just want to specify their transactions".

In Case 3 the consultant was thinking object oriented *" ... right from the start"* often because he was developing a prototype *" ... and that meant developing a reasonably functional prototype of every view or every screen. Some of the screens were quite complex and ... they formed the basis of the use cases and also the basis of the first cut of the production graphical interface. And if you're prototyping a graphical interface prototype and working through and developing use cases, you are talking about graphical objects and you naturally extend that and start talking about business objects as well."*

Case 4 saw the object-oriented nature of initial modelling in elicitation as linked to the concept of mental modelling *"I think that I do immediately start thinking of key objects during requirements gathering, not in any formal way, they just pop into one's head."*

In Case 5 the consultant did not believe he was looking at elicitation in object-oriented terms but in business terms *"No. Think 'business' all the way. (Compare OO thinking to 'Microsoft Word thinking' -- it's only when we come to tidy up and express ideas in Word that we think -- oh I'll try this as a list of bullet points, yes that looks right...) 'OO thinking' sounds weird ... it's really all about just thinking what the best thing to do is with the particular tools at your disposal."*

Case 6 believed that he thought in object-oriented terms from the start of the requirements engineering process but that other object-oriented analysts might not necessarily think that way. He saw it as a personal thing. *"When I do requirements documents I do [think OO]. If it's really requirements around something that already exists. If it's something new I start to think about what they are really after and then how to group together perhaps some of the requirements."*

So you are actually identifying objects at that stage – high level abstract objects?

Yes. A little. I certainly would. The majority of the BAs [Business Analysts] and some of the other more IT oriented people probably not. The aim of the document is not really to be object-oriented in any way. It is simply gathering what the requirements are."

Table 6.9 summarises the findings regarding the concept of analysts thinking in object-oriented terms during elicitation.

Case No	Analyst always thinking in object-oriented terms
Case 1	No, did not "think object-oriented" at all
Case 2	Yes, but it was a personal thing
Case 3	Yes, in order to produce a prototype
Case 4	Yes, key objects
Case 5	No, only business concepts
Case 6	Yes, but it was a personal thing

Table 6.9 Thinking object-oriented during elicitation

6.4.2 Use of a specific methodology

Another concept that emerged indirectly from necessary background questions was that of which methodology was used by an individual consultant and why. The questions which addressed this issue were:

- Which method(ology) is being used on this project?
- Why that method(ology)?
- Where does requirements specification fit within the methodology?
- Has this methodology been adopted across the organisation or only for this project?
- Comment on the advantages of the current methodology (if any) over previously used methods.

Case 1 used a proprietary method called Volere (Robertson and Robertson, 1997, Robertson and Robertson, 2001) which is a requirements-only methodology. The methodology is based on a template and the use of cards (called the requirements shell) to describe requirements. The template is a booklet that provides guidelines for the 24 types of requirements and the tasks which need to be undertaken during the process of requirements specification in order to develop both functional and non-functional requirements. Requirements cards, as described in section 5.3.3, are filled out in collaboration with the client/users during the requirements specification process. The methodology was described by the analyst as structured, easy to use and self-documenting.

Case 2 used an in-house methodology based on use cases and OMT models but not the whole Rumbaugh method and not the whole Jacobson use case method since Jacobson's method in full was considered *" ... too open-ended, too many decisions for clients to make."*

Case 3 also used an in-house method based on various methodologies that team members were familiar with or had found useful or successful in the past. Its main elements were OMT models, use cases and the Software Through Pictures (STP) case tool *"We sampled from methodologies that we were familiar with ... three or four people [who] were able to contribute to a methodology that picked up bits and pieces from a number of influences... They just all brought their biases and their interests and thoughts."*

Case 4 did not use any specific methodologies. The consultant professed to using a combination of entity-relationship and UML notation, ad hoc diagrams, pictures, use cases etc. *"I haven't been, let's say, an advocate of any particular methodology from start to finish...See I don't believe in methods as such ... What I talk about is a underlying concept rather than a methodology".*

Case 5 used no methodology at all. The development was based around ad hoc notes from whiteboards written up in loose-leaf and/or bound notebooks and *"Typed up 'processed' versions of these in an ad hoc, as needed, fashion. Many ideas get 'documented' in the code (use of [sic] Javadoc to include design ideas in code)."*

Case 6 used an in-house methodology based on UML notation. This consultant was the most discursive on the use of methodologies *"All of the diagramming is UML. UML is a notation, it's not a development methodology ... [Rational] is a formal development life cycle methodology about how to go through the whole process using UML as the documentation method ... I've not seen anybody use the big methodologies. I think there are two reasons. (A) If you buy the professional ones they charge too much, which is also why I think why everyone talks about Rational, although we tend to use a competitive product, Select, mainly because it's a little cheaper. And even then I don't have as many copies as I should have because it's so expensive. They are VERY high cost and if you put on the process flow modelling, all the methodology on top of that ... What happens is that the cost of setting up a developer starts to become prohibitive and there's no return on that so either you escalate the price of your product to cover that high cost or you hope you work for a multi billion dollar company that can afford to simply write cheques and say 'yeah we will spend all this money'. The other reason certainly why we tend to use our own methodologies which have short cuts and work arounds and all sorts of different things and why even methodologies where you are supposed to follow them (as in our case) there are odd documents missing and some are much shorter than they should be. There is simply not enough time to follow the whole box and dice and produce all of the documents. You produce those documents where, if you've got to get a user to sign off do those. Why, because that affects the bottom line and that's really what it's about."*

Four of the six cases used methodologies that were developed in-house rather than purchasing commercial or proprietary methodologies. Table 6.10 summarises the methodologies used in the six cases.

Case No	Methodology
Case 1	Volere, proprietary requirements-only methodology (free)
Case 2	In-house based on OMT and use cases
Case 3	In-house from team members experience based on OMT models, use case and Software Through Pictures (STP)
Case 4	In-house methodology based on ER, OMT, use cases, ad hoc diagrams, prototypes etc
Case 5	No methodology, ad hoc notes and diagrams were developed on a whiteboard and transferred to notebooks
Case 6	In-house based on ER, UML notation and use cases

Table 6.10 Use of a specific methodology

6.4.3 Evidence of opportunistic approaches to elicitation

Some of the literature discussing opportunistic approaches to analysis and design (Carroll and Swatman, 1997, Khushalani et al., 1994) was discussed in section 2.3.3 in Chapter 2. The concept of opportunistic approaches to elicitation in the sequential-case study emerged from questions about feedback in knowledge elicitation. The exploration of the amount of feedback and what triggered it during Case 3 produced the following statement *"Someone in a meeting or a discussion will say, 'Of course you know we only ever had one of these, and that will change' and you can say 'Ah! Test that against my understanding of what a customer, or event or a facility or whatever the abstraction is' and that might either verify or it might contradict it. If it verifies it you probably let things go and move on to the next point. If it contradicts it, you need to pick it up and mine that and get to the bottom of that."* This led to the formulation of the following question in subsequent interview scripts:

- Do you see the elicitation process as being sequential or does some piece of information trigger the need to explore some new feature or aspect?

On reflection Case 1 and Case 2 provided no evidence of opportunistic feedback and the participants were not asked explicitly about it. The analyst in Case 3 was the first participant to bring up the concept of opportunistic feedback in elicitation as evidenced by the quote above. He also linked opportunism in elicitation with mental models "*So you might carry round an event or an account or a customer object or something, you carry around a picture of how that is shaping in your mind.*

When explicitly asked about opportunism in elicitation the consultant in Case 4 greed that he operated that way as evidenced in the following exchange *" **Do you get that sort of feeling that as you are exploring the requirements that some user might say of course we need blah blah blah and then you say Ah! We should go and think about that and then come back maybe later to it.***
Certainly, yeah, yeah. The thing about gathering requirements is that you don't, a priori, you don't know what you're going to get so you've got to be opportunistic."

The consultant in Case 5 responded to the question directly and succinctly *"Very very much the things triggering further questions style."*

By Case 6 the question had become more embellished ***"Now the last question here is sort of again a bit about how you think and it's been prompted by talking to other consultants. And basically whether you could comment on the idea of this requirements gathering process, whether you see it as sequential or do you see it as more opportunistic where some piece of information that you get might trigger the need to explore some other aspect or feature and you tend to sort of jump around inside the domain building up the requirements picture."*** and brought the following response *"It's probably a mixture of the two. There is certainly some aim at trying to be sequential. I certainly try and do that when I start with the users, and try and build up a broad outline of everything first and then it becomes more opportunistic as working through...'Oh we need to*

expand this more', 'Ah, this has brought out a requirement that is in a completely different part of the system but I better go and resolve that now because that's how it's connected up'. "

Evidence of opportunism in requirements elicitation emerged in Case 3 and was subsequently reinforced in Cases 4, 5 and 6. Table 6.11 summarises the findings regarding analysts acting opportunistically during elicitation.

Case No	Evidence of opportunistic approaches to requirements elicitation
Case 1	No evidence
Case 2	No evidence
Case 3	Opportunism in elicitation emerged as worth exploring
Case 4	Evidence of opportunism in elicitation on explicit questioning
Case 5	Evidence of opportunism in elicitation on explicit questioning
Case 6	Evidence of opportunism in elicitation on explicit questioning but not from the beginning of the elicitation process

Table 6.11 Opportunism in knowledge elicitation

6.4.4 Summary

Specific findings related to the exploration of the opinions, beliefs and behaviours of the professional analysts engaged in object-oriented requirements engineering as posed in the research questions produced three major concepts which were reinforced in at least three other cases: analysts always thinking in object-oriented terms, the predominant use of in-house methodologies over commercial or proprietary methodologies and evidence of analysts acting opportunistically during requirements elicitation. The findings regarding these three concepts are summarised in Table 6.12.

	Analyst always thinking in object-oriented terms	In-house methodology	Opportunism in elicitation
Case 1		No, commercial	
Case 2	reinforced	revealed	
Case 3	reinforced	reinforced	revealed
Case 4	reinforced	reinforced	reinforced
Case 5		reinforced	reinforced
Case 6	reinforced	reinforced	reinforced

Table 6.12 Concepts related to the research questions but not directly related to the conceptual process model

6.5 Chapter Summary

This chapter has presented the sequential-case study findings relating to the three objectives:

- to determine the empirical validity of the initial conceptual process model presented in Chapter 4 of this thesis.
- to analyse the emerging characteristics of the evolving versions of the conceptual process model that are revealed and/or reinforced by the sequential-case study data presented in Chapter 5.
- to describe and discuss findings from the sequential-case study data that relate to the cognitive and social aspects of the research questions but do not relate directly to the conceptual process model.

The case study analysis has provided strong evidence (five out of six cases for all categories) for the empirical validation of the concepts in the initial conceptual process model.

The sequential-case studies provided data which allowed the initial conceptual process model to evolve through several versions which revealed and reinforced several concepts that were not present in the initial version of the conceptual process model. This analysis demonstrates findings grounded in the

examination of professional practice in requirements engineering as represented in the six sequential-case studies.

Finally, there was an analysis of the cognitive and social concepts that arose in the sequential-case studies based on the research questions which explore the opinions, beliefs and behaviours of the professional analysts engaged in object-oriented requirements engineering. The concept that elicitation was inherently object-oriented in object-oriented requirements engineering was explored in all six cases and in the five cases where object-oriented notations, techniques or methods were used this concept was reinforced. An investigation of the specific methodology used in each case revealed strong evidence for the use of in-house methodologies based on the analysts' own experience and personal preferences. The investigation of the use of iterative feedback for clarification in elicitation in the original conceptual model and interview questions revealed the use of opportunistic approaches to requirements gathering in four of the six cases.

The sequential-case study has contributed to understanding how object-oriented models are used in practice. The study has helped to clarify the situations in which certain types of models are used and why they are used in those situations. The study has produced a conceptual process model grounded first in the literature regarding object-oriented requirements engineering and then grounded in the professional practice of object-oriented requirements engineering as observed in the case studies. Chapter 7 of this thesis discusses the implications of the case study findings for practice and for the concepts of the conceptual process model.

Chapter 7

Implications of the Case Study Findings

7.1 Overview

This chapter discusses the implications of the case study findings in three main sections. The first section considers the implications of the findings for the structure of the conceptual process model initially proposed in Chapter 4 and refined in Chapter 6 of this thesis. The second section considers the implications of the findings for practice and for the study of professional practice in requirements engineering. The third section considers the implications of the findings for the education and training of professionals in requirements engineering. The chapter concludes with a discussion of the limitations of the research results.

7.2 Implications for the Conceptual Process Model

The case study analysis in Chapter 6 of this thesis has confirmed the empirical validity of the components of the initial conceptual process model proposed in Chapter 4 of this thesis. Relationships between the components of the conceptual process model, where applicable to the case data, were also confirmed by the case study findings. The findings also suggested additional components and features of

the conceptual process model that led to a revised conceptual process model (see Figure 7.1).

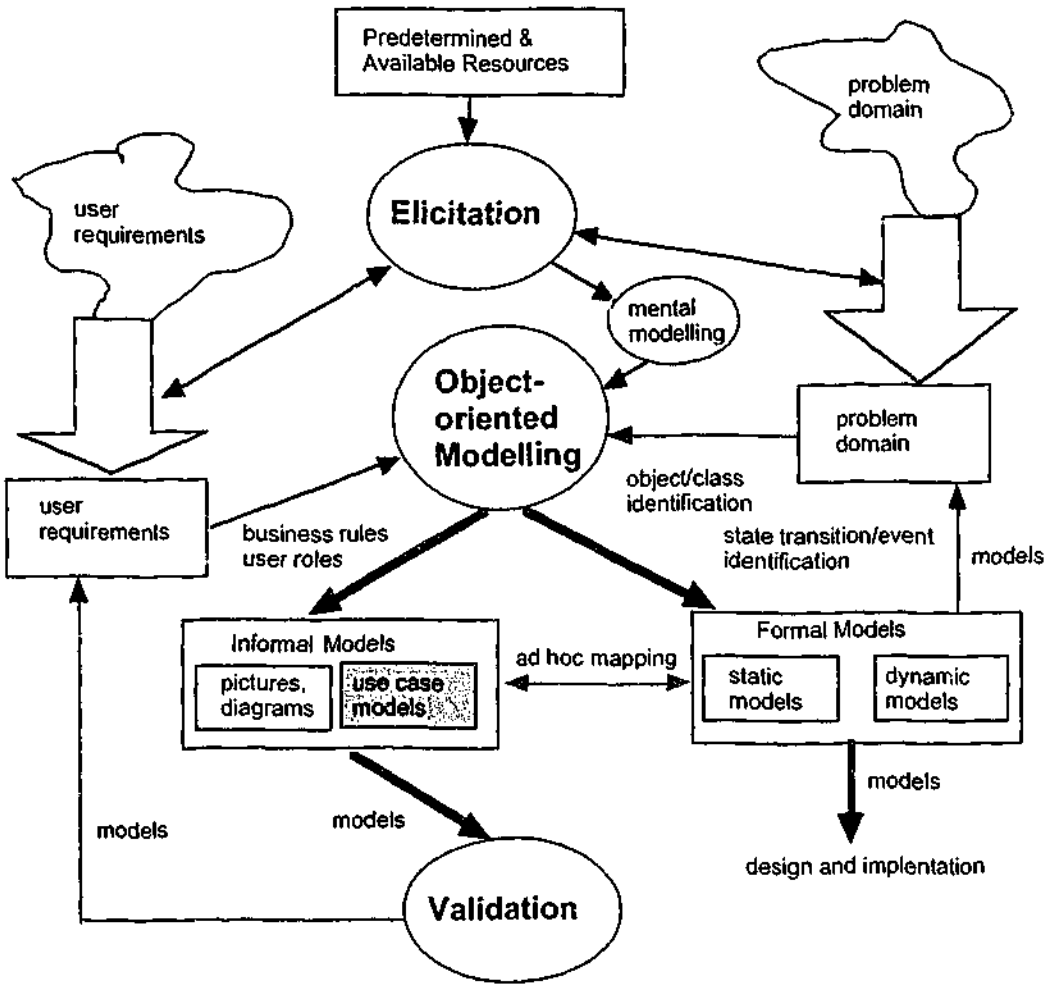


Figure 7.1 Final Revised Conceptual Model

For each of the fundamental concepts contained in the conceptual process model and analysed in section 6.2 of this thesis five of the six cases confirmed the initial concepts contained in the conceptual process model. It should be noted that it was a different set of five out of six cases for each concept as summarised in table 6.10.

- Five of the cases showed evidence of three distinct processes within the requirements engineering process. These five consultants explicitly performed elicitation (also called requirements gathering, informaton gathering and knowledge acquisition), modelling (also called requirements modelling or requirements representation) and validation (also called requirements testing or requirements validation).
- Five of the consultants reported that elicitation was always an iterative process and that elicitation relied on feedback.
- Five of the six consultants used static models for object-oriented requirements modelling and five consultants used dynamic models in object-oriented requirements modelling.

All of these fundamental components and relationships were retained from the initial conceptual process model. Additional components and features that emerged from the sequential-case studies are:

- In four of the six case studies a process of mental modelling occurred during elicitation. The consultants saw this mental modelling as natural to the elicitation and modelling processes and private to the individual analyst. This is now shown explicitly in the revised conceptual process model.
- In all cases except Case 5 there was evidence that the modelling process produced two types of models: informal models in the form of pictures, text and diagrams, and/or use cases, and prototypes, i.e. models that can be understood and explained without specific training; and formal models which are based on specific modelling notations such as entity-relationship modelling, OMT and UML modelling. These took the form of class models, object models and to a lesser extent interaction and event models that require training in order to be understood or explained. This separation of formal and informal models is also shown explicitly in the revised conceptual process model.
- The two types of models were used for different purposes. The informal models were used in the validation of the specification with the clients and users and the formal models were used internally within the analysis team and passed on to the design phase of the development. The use of the informal

models for feedback to the users for validation and the use of formal models for the design and implementation processes is also shown explicitly in the revised conceptual process model.

- There was also evidence of a less well-defined mapping of the informal models to the formal models. This appeared to be a two way process where each group of models was developed in parallel and where each group of models "informed" the other as they were developed by the analysts. The validation process was informal and formal models were not used in this process. This mapping is shown as a two-way communication path and the lack of definition is indicated by a dotted line in the revised conceptual process model.

These findings indicate that analysts believe that users or clients find formal models much too complex, both conceptually and technically, to understand and that the use of informal models such as rich pictures, diagrams and use cases, particularly use case scripts which are closer to natural language models, are perceived to be better models for communicating and validating specifications with clients. Further research is necessary to determine the validity of this perception.

The revised conceptual process model now embodies all the concepts of the initial conceptual process model which were proposed in Chapter 4 based on the literature regarding object-oriented requirements engineering together with the concepts that emerged from the sequential-case studies presented in Chapter 5 and analysed in Chapter 6 of this thesis. This revised conceptual process model provides a theoretical representation of the object-oriented requirements engineering process grounded in the literature and in professional requirements engineering practice.

7.3 Implications for Requirements Engineering Practice

The basis for the perceived need for both informal and formal models in object-oriented requirements engineering as found in this study may lie in the fact that the requirements engineering process is fundamentally a social process involving

two main groups: the users/clients and the professional consultants (Urquhart, 1998, Loucopoulos and Karakostas, 1995, Macaulay, 1996). This section presents implications for both object-oriented and non object-oriented professional requirements engineering. These implications relate directly to, and are based on, the findings of the six sequential-case studies presented in Chapter 6 of this thesis. It is not claimed that the implications discussed here are new or exhaustive, rather that the findings from this research project strengthen the idea that requirements engineering is a social, creative, and cognitive process (Darke and Shanks, 1997, Flynn and Warhurst, 1994, Galal and McDonnell, 1998, Hawryszkiewycz, 1994, Lockwood and Lamp, 2000, Urquhart, 1998). A major finding of this research project is the use of informal models for representing the requirements to users because they are easier to understand as distinct from the formal models built by the analyst and based on specific notations which are passed on to the developers of the design and implementation phases of system development. This finding has the following implications for requirements engineering practice. Each point follows from the one before:

- Requirements engineering is a social process and this social process requires *understanding* by all parties to reach *agreement*. Understanding requires communication skills and agreement requires negotiation skills.
- The facilitation of understanding and agreement requires creative modelling skills on the part on the analyst to produce understandable informal models. These models are developed during elicitation, refined during modelling and used for validation of requirements before sign-off or agreement to go ahead with design and implementation.
- Creative informal modelling as demonstrated by the analysts in this study relied on cognitive skills including
  - abstraction and mental modelling and,
  - problem-solving and reasoning skills particularly analogical reasoning skills (defined in section 2.3.2 as abstracting a solution strategy from one problem and relating that information to a new problem) on the part of the analyst.

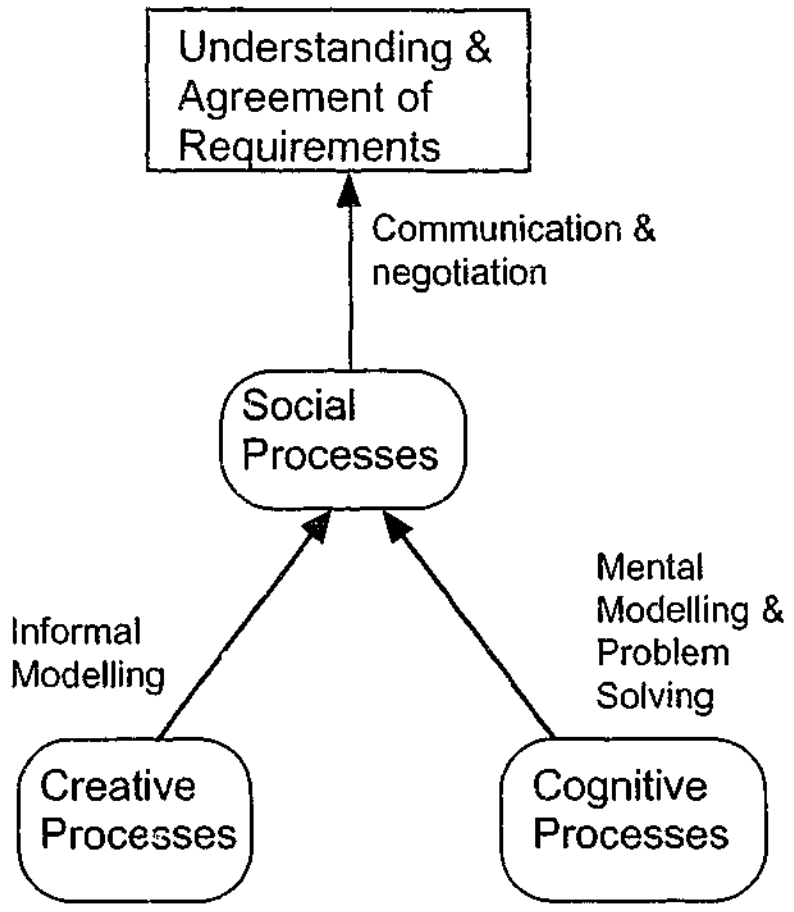


Figure 7.2 A theoretical model of the contribution of social, creative and cognitive processes to requirements engineering

These concepts and their relationships are represented in thetheoretical model shown in Figure 7.2. This model contains features for defining a theoretical model as defined by Dubin (Dubin, 1976) and Bacharach (Bacharach, 1988) i.e. the interactions or relations between defined units or concepts within a set of boundaries or constraints depicting a limited portion of the world.

In this model the main social goal of successful requirements engineering is to achieve agreement and understanding about requirements between users/clients and the professional developer or development team. The achievement of this goal depends on three processes. The social process involves the users and the analysts in communication and negotiation which brings about the understanding and agreement. This social interaction is influenced by the professional input of the analyst in the role of problem-solver and mental modeller. Further, the analyst also has to express the solutions to the problems and the models arrived at in his/her mind in a concrete manner which faciltates the understanding and agreement. This

creative process involves the development of informal models (such as diagrams, simulations, animations or textual explanations) that can be understood and discussed by the users and analysts in their social communications and negotiations. These implications are discussed further in the following two sections.

7.3.1 Requirements Engineering as a Social Process

The social aspects of the requirements engineering process have been well-documented (Urquhart, 1998, Macaulay, 1996, Loucopoulos and Karakostas, 1995, Galal and McDonnell, 1998).

Viewing the requirements engineering process as a social process implies the following: If the *product* of the requirements engineering process is the specification document on which the design and implementation of the system is based, then this product has to be agreed upon by both parties. That is, the specification needs to be validated as correct or acceptable from both points of view - the *formal* or consultant's point of view and the client's *informal* point of view.

The perception of the analysts in this study seems to be that for this agreement to take place there needs to be two types of models: informal models for communicating the specification to the user for information and validation; and formal models developed by the analyst team to pass on to the design and implementation team.

As discussed in Chapter 2 of this thesis it has been generally recognised that many of the errors that lead to costly maintenance and/or failure of information systems can be traced to ommissions, inconsistencies and ambiguities in the initial requirements specification. If, as the findings of this research project suggest, the models used for validation of the specification with the clients are different to the models used in design and implementation, then this may indicate one of the areas where these inconsistencies, omissions and ambiguities might arise. Recognising

and understanding this issue requires further research and provides a step towards building the right tools and techniques to assist the requirements engineering process.

7.3.2 Requirements Engineering as a Creative and Cognitive Process

As with many professional activities involving analysis and design (Schön, 1983, Khushalani et al., 1994, Johnston, 1999, Galal, 1998), object-oriented requirements engineering and requirements engineering in general can be considered to be a creative process particularly on the part of the requirements engineer or the analyst undertaking the requirements specification. Recently work has been undertaken to develop tools and techniques to support the requirements engineering process (Haywood and Dart, 1999, Cybulski and Reed, 1999). For this kind of development of requirements engineering support environments to take place in the appropriate context, research such as the research undertaken in this research project is needed to provide the theoretical and contextual foundations for the development of new tools and techniques. That is, there is a need to investigate various aspects of the requirements engineering process in practice, as in this project, in order to understand the process and how it is currently being carried out by practising professionals so that tools and techniques can be developed to support the process.

The case studies showed evidence of recognition on the part of the practising professionals that they had to be able to model or represent what the users wanted in some diagrammatic form. The findings suggest all of the analysts who used formal object-oriented notations such as OMT or UML or other formal notations like entity-relationship diagrams would not use diagrams based on these notations with the users or clients because they believed the users would not understand them. For all of the analysts in this study this meant that they had to find creative solutions to the representation problem. The creative solutions provided by the analysts are the informal models as defined in this thesis (section 6.3.3) and were considerably diverse: simple use cases, adhoc diagrams, rich pictures, animations, PowerPoint simulations and text based explanations. Each analyst had his/her

own creative approach to informal user modelling. There is enough evidence provided in the case studies to imply that this creative approach to user modelling is common in professional requirements engineering practice. This suggests that the variety and use of such informal models should be systematically described in detail, which would be useful to professionals, educators and students (see below).

As described in section 2.3.3, closely related to the creative aspects of requirements engineering are the cognitive aspects of requirements engineering as evidenced in the findings of this research project. Requirements specification can be considered as a high level cognitive process (Khushalani et al., 1994, Schön, 1983, Mayer, 1992, Sutcliffe and Maiden, 1992, Gick and Holyoak, 1980). In four of the six cases reported in this thesis requirements specification involved mental modelling during the transformation from elicitation to concrete models for design and implementation. The consultant examined and discussed problems encountered with the users during elicitation and then modelled the solutions to those problems in their own minds before committing to concrete models used for design and implementation. This mental modelling process appeared to involve abstraction and analogous reasoning as well as problem-solving activity.

7.4 Implications for Education and Training

Typical undergraduate courses in information systems or related disciplines involve some exposure for students to systems analysis methodologies, techniques and tools. Often students are required to participate in a project where the principles of systems analysis can be applied to an example of an industrial or commercial style systems development project. The challenge for academics designing these courses or writing textbooks to accompany these courses is often how to relate theory and project work to real professional practice. Successful programmes in these areas have provided projects with real clients, project team environments (Keen et al., 1998, Lamp and Lockwood, 2000, Lockwood and Lamp, 2000) and more recently studio-based environments (Carbone et al., 2000).

The findings from the case studies in this research project have several implications for education and training. Based on the findings presented in this thesis, courses seeking to provide realistic commercial project environments should include the following elements and ideas:

- There are many tools and techniques for requirements analysis and specification and as this research project has shown:
  - In practice analysts often develop their own in-house methodologies based on diverse tools and techniques rather than adhere to a single prescribed or commercial methodology
  - Many professional analysts build their own "conceptual toolkit" or personal methodology by trying out and adapting those techniques and tools that suit their way of thinking and their way of interacting with clients and the particular projects that they are working on.
- There are many models for representing requirements and as this research project has shown:
  - Users may not understand formal notations like ER, OMT and UML diagrams
  - Some professional analysts develop informal models based on adhoc diagrams, rich pictures, animations, PowerPoint simulations, text based explanations, simple use cases, and simple use cases for explaining requirements to users/clients.
  - Informal models which are not based on formal notations like ER, OMT and UML are often the basis for agreement and sign-off for requirements specifications

The implication for education and training from these ideas (and suggested by the case study findings) is that for students to be able to undertake a major project they need to be encouraged to build their own conceptual toolkit after being exposed to as many tools and techniques as possible. This also implies that students should be encouraged to experiment with and develop some informal modelling techniques for communicating requirements to users/clients. This also concurs with the work of Haywood and Dart (1999) which suggests that there are many different

modelling methods and notations available and that some are more appropriate for certain types of projects than others.

7.5 Limitations of Research Results

The research method used in this thesis produced rich qualitative data which was analysed using interpretive, qualitative techniques. The conceptual process model presented in Chapter 4 of this thesis was grounded in the literature with regard to object-oriented requirements engineering. The construction of the initial version of this model and its subsequent refinement, based on sequential-case studies actively seeking information based on using leading questions to clarify prior findings, can be described as subjective. This subjective nature means that there is potential for biased interpretations on the part of the researcher (Neuman, 1994, Galliers, 1992, Shanks et al., 1993).

The value and quality of research using the conceptual study approach and based on subjectively developed models is difficult to assess. Possible criteria for determining the value of this research include:

- the internal consistency of the theoretical concepts proposed in the conceptual process model (Figure 7.1) and the theoretical model (Figure 7.2) described above
- the degree to which the conceptual model was grounded in the literature in the first place
- the degree to which the final conceptual model is grounded first in the existing literature and then in the findings from practice. This is related to the degree of theoretical saturation (Eisenhardt, 1989) reached based on six sequential-case studies
- the degree to which the theoretical model is grounded in the literature and the findings from practice.

The strength of the validity of the findings can be examined by addressing the following specific issues:

The conceptual process model was subjectively and selectively constructed but it can be argued that it was well-grounded in the literature.

The number of case studies in this project was six and was not intended to provide quantitative or statistical data. The nature of the research project and research questions suggested that the data would have to be qualitative and the analysis interpretive. Qualitative interview-based data was therefore gathered in these case studies. The case study findings suggest that a level of theoretical saturation (Eisenhardt, 1989) was reached since no new categories emerged after Case 4. However, additional cases studies would add to the evidence gathered in the six cases examined in this research project and would further strengthen the results if confirmatory evidence were found.

The subjective nature of the data collection which was based on semi-structured interviews with practising professionals. The data is based on perceived behaviour on the part of the analyst/participants. There was no attempt to collect data based on demonstrated or observed behaviour. There is scope for future research in this area.

The interpretive nature of the data analysis which was based on categorised transcription. The rich qualitative nature of the data collected meant that an interpretive approach to analysis was required. The limitations of interpretive analysis were recognised and attempts were made to minimise the problems associated with interpretive analysis by using a structured top-down categorisation approach where initial categories for investigation and analysis were formulated based on the conceptual process model and the research questions.

The top down nature of the categorisation of the data provided significant structure to the collection and analysis of the data but is constrained by the need to make initial assumptions about the data in order to formulate the initial categories.

The in-situ parts of the data analysis inherent in the sequential-case study approach allowed findings to influence subsequent data collection. This

potential limitation is bound to the research approach of using sequential-case studies to actively explore emerging categories and concepts which arise in one case in subsequent cases. From the perspective of exploratory research this potential limitation could be regarded as a necessary and defining characteristic of the research approach.

The cumulative nature of the data collection means some data could be missed in earlier cases. This limitation is based on the concept that interview transcripts in the sequential-case approach grow as the number of cases increases. This is because each case is exploring additional emerging concepts until theoretical saturation or sufficient closure is reached.

Although the nature of this research approach means that results can never be statistically exhaustive, it can be suggested that evidence for the conceptual process model and other findings which have addressed the research questions has been found and documented in a structured and systematic way. It can therefore be argued that this research project has contributed to the body of knowledge about object-oriented requirements engineering.

7.6 Summary

This chapter has identified the implications of the case study findings for the theoretical concepts proposed in the conceptual process model in Chapter 4 of this thesis. Implications of the case study findings for the use of object-oriented models in requirements engineering practice have also been identified and discussed. This discussion led to a theoretical model of the processes and influences involved in the object-oriented requirements engineering process with respect to the development and use of informal models for reaching understanding and agreement between users and consultants. The case study findings also suggest a need to examine education and training methods for requirements engineers and systems analysts with respect to developing diverse approaches to the use of models, methodologies, techniques and tools for elicitation, modelling and validation of user requirements. The chapter has also identified some of the limitations of the findings in this research project.

Chapter 8

Conclusion and Future Work

8.1 Summary of this Research Project

Few empirical studies have been published which investigate how practising professionals use object-oriented methods and models in requirements engineering. This research project has addressed this gap in the literature and has described the findings from a set of six case studies which examined the use of object-oriented models in professional requirements engineering practice.

This research project has sought first to define and describe the key theoretical concepts in object-oriented requirements engineering, particularly with respect to the use of object-oriented models to support the requirements engineering process. An extensive literature analysis and review of existing approaches to and models of requirements engineering in general and object-oriented requirements engineering in particular was undertaken. This revealed that, although there are several conceptual frameworks or theoretical models of requirements engineering in general (Darke and Shanks, 1996, Loucopoulos and Karakostas, 1995, Macaulay, 1996, Pohl, 1994), very little research has been

directed at developing a specifically object-oriented conceptual model or framework. A substantial contribution of this research project has been to develop such a conceptual process model. The empirical validity of the concepts in the model was evaluated by examining object-oriented requirements engineering practice using a sequential-case study method

The two major research components of this project can be summarised as follows:

The Conceptual Study

This research component established a well-grounded conceptual understanding of object-oriented requirements engineering. A conceptual process model was developed from an analysis of the existing literature using the conceptual study research method (Shanks et al., 1993, Galliers, 1992). The conceptual process model identifies and describes the key concepts, processes and relationships of object-oriented requirements engineering as described in the literature. The conceptual process model is an important contribution to theory within object-oriented requirements engineering.

The Multiple Sequential-Case Study

The research questions proposed in Chapter 3 of this thesis and the conceptual process model proposed in Chapter 4 and refined in Chapter 6 provided the basis for the design and conduct of the sequential-case study in Chapter 5. The multiple sequential-case study comprised six cases which examined the use of models in object-oriented requirements engineering in practice in order to determine the empirical validity of the conceptual process model. The conceptual process model together with the research questions provided the basis for the interview scripts used in the sequential-case study. The sequential-

case study also sought to further refine the conceptual process model by actively exploring concepts that were evident in each case in subsequent cases.

8.2 Research Results

This thesis makes several contributions to the theory and understanding of the use of object-oriented modelling methods in requirements engineering.

The case study results

- confirmed the empirical validity of the concepts and relationships embodied in the initial conceptual process model
- revealed new characteristics of the evolving versions of the conceptual process model as they emerged during the conduct of the sequential-case study.
- provided research findings of interest concerning reuirements engineering that were independent of the conceptual process model.

The case study findings also answered the three major research questions. The aggregation of the answers to these questions address the broad research theme proposed in Chapters 1 and 3 of this thesis:

"How are object-oriented modelling methods used by practising professionals in the process of requirements engineering?"

This section outlines research findings in terms of answers to the research questions. The first major research question

Is elicitation influenced by the use of object-oriented modelling methods?

was addressed in the case study by responses to the interview question:

Is knowledge elicitation specifically object-oriented, i.e. do you think "OO" from the start?

The concept of analysts thinking in specifically object-oriented terms was confirmed by the findings. Four of the six analysts believed that they were always thinking in object-oriented terms while carrying out elicitation of user requirements. The case study data provided evidence that elicitation is specifically object-oriented.

The second major research question

When, how and for whom is object-oriented modelling undertaken?

was addressed in the case study by responses to the following interview questions:

Which (how many) models are produced during specification?
Do you produce class models, use case models or interaction models?
Would you categorise models as static or dynamic?
Who uses them?
Who are they produced for?
Which models, if any, are shown to the user?
Which models are used internally by the development team?

The findings provided evidence of the use of two different types of models. Informal models were used to represent the requirements in terms the users could understand. These models took the form of adhoc diagrams, pictures, animations, PowerPoint mock-ups, use case scripts etc. Formal models based on specific modelling notations were only used within the analysis team or with the professional developers. The case study data revealed that object-oriented models (written in specific notations) are only used within the analysis and/or

system development team. Object-oriented models based on formal notations are never, or rarely, developed for or shown to users.

The third major research question:

How is validation performed on object-oriented models?

is addressed in the case study by responses to several interview questions:

Which (how many) models are produced during specification?
Who uses them?
Who are they produced for?
Which models, if any, are shown to the user?
Which models are used internally by the development team?
Do you think it is necessary to validate the specification once the models have been produced?
When does the validation process start?
Could you comment on the role and/or importance of use cases in the specification process?

The case study findings provided evidence that no formal processes were used for the validation of the requirements with users in practice. None of the analysts used any mathematical proofs or formal methods in validation. All analysts used walkthroughs to some extent, together with the informal models described above for validation.

The case study data provided evidence that validation of object-oriented models used in requirements specification is not generally carried out using formal methods. Rather the validation process is based on informal models representing the formal models or mapped from the formal models in an

adhoc way. The informal models are the models used in walkthroughs with the users for validation and sign off of a requirements specification.

8.3 Future Work

Several issues arising from the research reported in this thesis have been identified for further research and investigation. The following sections describe six areas of future research including suggestions for specific research projects to address these areas.

8.3.1 Mapping Formal Models to Informal Models

As discussed in Chapters 6 and 7 of this thesis, one of the major findings of this research is the identification of the use of separate models for the representation of requirements depending on who is the target audience for the models. No formal models are developed for communicating requirements to users and clients. Formal models based on specific notations are used only with trained professionals, usually the analysis and/or development team. These separate models are explicitly represented in the final version of the conceptual process model in Figure 7.1. The least defined concept in the final conceptual process model is the mapping of the informal models to the formal models and vice versa. The most common description of this mapping process given by the analysts in the case study was that it was a "natural" or "automatic" process for which professional analysts had to develop the skills. This aspect of the modelling process may be further investigated using an ethnographic study or by using protocol analysis techniques where practising analysts are asked to describe what they are thinking and doing while they are explaining models to users or other analysts.

8.3.2 Comparing perceived and demonstrated behaviours in professional requirements engineers

The sequential-case study carried out for this research project included post hoc interviews with individual analysts describing the approaches they used in a single system development project. This form of study captured rich data about the perceived behaviours, opinions, beliefs and recollections of professional analysts. This data, as argued in this thesis, provides a useful perspective on professional requirements engineering practice. Further investigation could be undertaken using observational studies of analysts as they are working on system development projects in progress. These studies could provide rich data about demonstrated behaviour rather than perceived behaviour. "Perceived" refers to the activities and protocols the analyst *believes* he/she is undertaking whilst "demonstrated" refers to the activities and protocols that the analyst can be *observed* to be undertaking.

8.3.3 Migration to Object-Oriented Systems

One of the motivations for this research project is that many organisations are choosing to move from traditional software development methods to object-oriented methods for the development of information systems. Understanding why and how organisations are performing this migration would add to the body of knowledge about information systems development methodologies. For such a research project, the interest would focus on the organisation's motivation for change: why and how they migrate to object-oriented methods and the specific methods that they choose. Some potential research questions could be:

- Why has this organisation chosen to use object-oriented methods for system development?
- Has this organisation used traditional methods in the past? Which methods?

- Which object-oriented methodology has been chosen, and why?

In addressing these and other issues concerning the migration to object-oriented methods, such a research project could use case studies and/or surveys of organisations that have undertaken or are currently undertaking the migration process.

8.3.4 Cognitive Processes in Requirements Engineering

The research project reported in this thesis attempted to gather evidence in order to better understand behaviour and practices of professional system developers and requirements engineers. Further information about the cognitive processes involved in requirements engineering and systems development would contribute to the body of knowledge about requirements egineering and system development practice. In order to understand *what* successful system developers do, *how* they do it, *why* they do what they do and *when* they do what they do, further investigation is needed. Key areas include:

- the application of past experience by professional analysts to new problems
- investigation of commercial-scale information system development projects and the methods used by individual analysts and teams of analysts in managing these projects
- investigation of the techniques and methods used by practising professionals, including work patterns (time slicing or "multi-tasking"), action plans and other work activities

Studies of practising professionals working on large projects over the life of the project would need to be undertaken in order to fully explore these areas. This is because studying the behaviour of system developers over short time periods on a single task may not provide an adequate understanding of developers'

choices of, and success with, particular methods. Longitudinal studies may provide a better understanding. The sample sizes of both analysts and tasks used in many published studies of developers' behaviours may provide limited indicative results. More general, and possibly more useful, results may be obtained using larger sample sizes where possible.

The application of past experience to new problems

Several studies have been undertaken to measure differences in problem solving and system specification activities using novice (student) subjects in short sessions working on one problem (Chaiyasut and Shanks, 1994, Morris et al., 1996, Sutcliffe and Maiden, 1992). Even where "experts" are used these experts are often senior students who have passed specified courses (Guindon, 1990, Chaiyasut and Shanks, 1994). However, these experts can still be considered to be novices. The performance of "true" experts or experienced professional system developers needs to be investigated since successful use of a modelling or development method may only be developed over time. A method which may at first appear to be difficult to use may become easier and more effective over time. Also, a more experienced analyst may use more of a method's advanced features or may more effectively select appropriate techniques within the method during the requirements engineering process.

Gaining experience during a complex project or several projects may require the building of a "conceptual toolkit" or set of reusable tools and techniques which in turn may make a method more effective long-term than another method. Potential issues are:

- Does an analyst's preferred method become more effective the more often it is used?
- Does that method become easier to use the more often it is used?

- Does that method produce a better product the more often it is used?
- Does that method assist an analyst to reuse concepts and frameworks which have been successful in the past?
- Have analysts who have learnt new methods (by choice or necessity) found new techniques and tools that are better than the ones they used previously?

Investigating commercial-scale system development projects

Existing studies of system development have rarely involved large projects (Chaiyasut and Shanks, 1994, Guindon, 1990, Morris et al., 1996, Sutcliffe and Maiden, 1992, Vessey and Conger, 1994). Most studies focus on time periods of hours (at most) of work on a single, simple task (specification, modelling or programming problem). Studies of larger scale problems involving complete specifications of complex information systems need to be undertaken. Potential research questions associated with this issue are:

- Are some development methods better for small problems than other methods?
- Are some development methods better for large problems than other methods?
- Do experienced analysts use different development methods (or parts of methods) for different problem sizes?

Effects of work practices on the effectiveness of system development methods

Practising system developers tend to be working on more than one task or even more than one project at a given period in time. The integration of a specific task with other work activities may have an effect on the effectiveness of a chosen or preferred system development method. It may be necessary for developers to "multitask" or "time-slice" their work activities so that attention

is given to various tasks at different times. This type of work pattern may impact on the application of a new method or may have an impact on the choice of method for a particular phase of system modelling or specification. The actual choice of time-slice (when and how long) may also be an important factor. Of interest is how quickly a developer can resume productive work using a method after having spent some time away from the task whilst engaged in other activities. Potential research questions associated with this issue are:

- Are some development methods more effective for continuous work patterns than time-sliced work patterns?
- Do analysts use development methods that are easy to come back to for short time spans because of work patterns?
- Do analysts avoid development methods that require "relearning" after a period away from a problem?

8.3.5 Applying Usability Metrics to the use of Information System Development Methodologies

Evaluating the effectiveness of different system development methodologies is difficult. Usability metrics are used to measure the usability of human-computer interfaces and software systems. It should be possible to apply usability metrics to system development methodologies.

Usability is defined as

"...a measure of people's ability and motivation to use a product in practice. This definition must be applied in context. It depends on the user, the task to be done and the environment in which the product is used. The environment, in particular, can be complex; it is a combination of physical, social and technical

factors. It is meaningless to talk about the 'usability of a product' alone; a product may be perfectly usable but only by the product's developers."
(Dresner, 1996)

The MUSiC project (part of the European Community ESPRIT group of projects) identifies four kinds of metrics for human-computer interfaces (Hawryszkiewycz, 1994, MUSiC, 1993):

- *analytical metrics*, which can be directly described;
- *performance metrics*, which include things like the time used to perform a task, system robustness or how easy it is to make the system fail;
- *cognitive workload metrics* or the mental effort required of the user to use the system. It covers aspects such as how closely the interface approximates the user's mental model; and
- *user satisfaction metrics*, which include such things as how helpful the system is or how easy it is to learn.

A requirements specification method can be considered to be an "interface" between the developer and the developing system, so an adaptation of the metrics above could be useful in determining the effectiveness or "usability" of a specification method.

In examining this issue, usability would be addressed at a *meta-level*. The focus would be on the usability of one or more methodologies rather than the usability of the product that is produced by that methodology. i.e. the usability of the *process*, not the product of that process.

As outlined above, usability metrics for Human Computer Interface (HCI) design address four areas and could be adapted to provide usability metrics for information system development methods (ISM) as outlined below:

1. analytical metrics - what can be directly described

e.g. HCI: "Is all information needed by the user available on the screen(s)?"

e.g. ISM: "Is all information needed by the analyst to model and specify the system available from the user and problem domain?"

2. performance metrics - timing, robustness

e.g. HCI: "How long does it take to perform a task?" "How easy is it for the interface to fail?"

e.g. ISM: "How long does it take to perform a task?" "How easy is it for the method to fail to produce a useful specification, design or implementation?"

3. cognitive workload metrics - matching the interface with the user's mental model

e.g. HCI: "Does the interface match the user's mental model?"; "What is the mental effort required by the user to use the interface?"

e.g. ISM: "Does the specification method match the developer's and user's mental model?"; "What is the mental effort required by the user and developer to model and specify the system?"

4. user satisfaction metrics - satisfaction with, and willingness to use the interface

e.g. HCI: "How easy is the interface to learn and use?"; "How helpful is the interface and its prompts?"

e.g. ISM: "How easy is the method to learn and use?"; "How helpful is the method and its tools and techniques?"

8.4 Conclusion

This thesis has described the first significant study of the use of object-oriented modelling methods in requirements engineering practice. The thesis makes

contributions to the theory of object-oriented requirements engineering and requirements engineering in general in four specific areas:

- the development and refinement of a conceptual process model representing the key concepts and their relationships in the object-oriented requirements engineering process. This model was initially grounded in the existing requirements engineering literature and then refined and emprically validated by in-depth case studies of current practice.
- the proposition of a theoretical model based on the findings from the case studies which assists in explaining current requirements engineering practice and the implications of the findings for practice and for training in requirements engineering
- the formulation and demonstration of a research approach specifically tailored to this research project but which provides a useful research framework for similar theory building research, particularly in information systems.
- a platform for extensive further research programmes which are either directly related to this work or that flow from findings associated with this work.

References

Avison, D., Lau, F., Myers, M. and Nielsen, P. A. (1999) Action Research, *Communications of the ACM*, 42, (1) pp. 94-97.

Avison, D. E. and Fitzgerald, G. (1995) *Information Systems Development: Methodologies, Techniques and Tools*, McGraw Hill.

Bacharach, S. (1988) Organizational Theories: Some Criteria for Evaluation, *Academy of Management Review*, 14, (4) pp. 496-515.

Banville, C. and Landry, M. (1989) Can the Field of IS be Disciplined?, In *Information Systems Research: Issues, Methods and Practical Guidelines* (Ed. R. D. Galliers) Blackwell Scientific Publications, Oxford, pp. 61-88.

Bartlett, F. C. (1958) *Thinking*, Allen and Unwin, London.

Baskerville, R. and Wood-Harper, A. T. (1998) Diversity in information systems action research methods, *European Journal of Information Systems*, 7, pp. 90-107.

Baskerville, R. L. and Wood-Harper, A. T. (1996) A critical perspective on action research as a method for information systems research, *Journal of Information Technology*, 11, (3) pp. 235-246.

Beck, K. and Cunningham, W. (1989) A Laboratory for Teaching Object-Oriented Thinking, *SIGPLAN Notices*, 24, (10) pp. 1-6.

Benbasat, I. (1984) An Analysis of Research methodologies, In *The Information Systems Research Challenge* (Ed. F. W. McFarlan) Harvard Business School Press, Boston, pp. 47-85.

Benbasat, I., Goldstein, D. K. and Mead, M. (1987) The Case Research Strategy in Studies of Information Systems, *MIS Quarterly*, 11, (3) pp. 369-386.

Bjorner, D. and Jones, C. B. (1987) *The Vienna Development Method: The Meta-Language*, Springer Verlag, Berlin.

Blaha, M. and Premerlani, W. (1998) *Object-Oriented Modeling and Design for Database Applications*, Prentice-Hall, Upper Saddle River, NJ.

Boehm, B. W. (1976) Software Engineering, *IEEE Transactions on Computers*, 25, (12) pp. 1226-1241.

Boehm, B. W. (1981) *Software Engineering Economics*, Prentice-Hall, Englewood Cliffs.

Boehm, B. W. (1984) Verifying and Validating Software Requirements and Design Specifications, *IEEE Software*, 1, (1) pp. 75-88.

Boehm, B. W. (1988) A spiral model of software development and enhancement, *IEEE Computer*, 25, (5) pp. 61-72.

Boehm-Davis, D. and Ross, L. (1992) Program design methodologies and the software development process, *International Journal of Man-Machine Studies*, 36, pp. 1-19.

Booch, G. (1987) *Software Engineering with Ada*, Benjamin/Cummings, Menlo Park, CA.

Booch, G. (1991) *Object-Oriented Design*, Benjamin/Cummings, Menlo Park, CA.

Booch, G. (1994) *Object-Oriented Analysis and Design with Applications*, Benjamin/Cummings, Redwood City.

Booch, G., Rumbaugh, J. and Jacobson, I. (1999) *The Unified Modeling Language User Guide*, Addison-Wesley, Reading, MA.

Bruner, J. S., Goodnow, J. J. and Austin, G. A. (1956) *A study of thinking*, Wiley, New York.

Bubenko, J. and Wangler, B. (1991) Research Directions in Conceptual Specification Development, In *Conceptual Modelling, Databases, & CASE: An integrated view of information systems development* (Eds, P. Loucopoulos and R. Zicari) Addison-Wesley, Reading, pp. 389-412.

Budd, T. (1997) *An Introduction to Object-Oriented Programming*, Addison-Wesley, Reading MA.

Carbone, A., Lynch, K., Arnott, D. and Jamieson, P. (2000) Adopting a studio-based education approach into Information Technology, In *Proceedings of Fourth Australasian Computing Education Conference (ACE2000)*, Melbourne.

Carroll, J. and Swatman, P. (1997) How Can the Requirements Engineering Process be Improved?, In *Proceedings of Eighth Australasian Conference on Information Systems*, (Ed. D. J. Sutton) University of South Australia, Adelaide, South Australia, pp. 458-470.

Carroll, J. M., L.L., D. and Swatman, P. A. (1998) Using Case Studies to Build Theory: Structure and Rigour, In *Proceedings of Ninth Australian Conference on*

Information Systems, University of New South Wales, Sydney, Australia, pp. 64-76.

Cavaye, A. (1996) Case study research: a multi-faceted research approach for IS, *Information Systems Journal*, 6, (3) pp. 227-242.

Chaiyasut, P. and Shanks, G. (1994) Conceptual Data Modelling Process: A Study of Novice and Expert Data Modellers, In *Proceedings of First International Conference on Object Role Modelling*, Magnetic Island, Australia, pp. 310-333.

Checkland, P. (1981) *Systems Thinking, Systems Practice*, Wiley, Chichester.

Checkland, P. and Scholes, J. (1990) *Soft Systems Methodology in Practice*, Wiley, Chichester.

Chen, P. (1976) The entity-relationship model:toward a unified view of data, *ACM Transactions on Database Systems*, 1, (1) pp. 9-36.

Chua, W. F. (1986) Radical Developments in Accounting Thought, *The Accounting Review*, 61, pp. 601-632.

Coad, P. and Yourdon, E. (1990) *Object-Oriented Analysis*, Yourdon Press/Prentice Hall, New York, USA.

Coad, P. and Yourdon, E. (1991) *Object-Oriented Analysis*, Yourdon Press/Prentice-Hall, Englewood Cliffs, NJ.

Coleman, D., Arnold, P., Bodoff, S., Dollin, C., Gilchrist, H., Hayes, F. and Jeremaes, P. (1994) *Object-Oriented Development: the Fusion Method*, Prentice Hall, Englewood Cliffs, NJ.

Craik, K. J. W. (1943) *The Nature of Explanation*, Cambridge University Press, Cambridge.

Cybulski, J. and Reed, K. (1998) Computer Assisted Analysis and Refinement of Informal Software Requirements Documents, In *Proceedings of Asia-Pacific Software Engineering Conference (APSEC'98)*, Taipei, Taiwan, pp. 128-135.

Cybulski, J. and Reed, K. (1999) Automating requirements refinement with cross-domain requirements classification, In *Proceedings of Fourth Australasian Conference on Requirements Engineering*, Sydney, Australia, pp. 131-145.

Darke, P. and Shanks, G. (1996) Stakeholder Viewpoints in Requirements Definition: A Framework for understanding Viewpoint Development Approaches, *Requirements Engineering*, 1, pp. 88-105.

Darke, P. and Shanks, G. (1997) Managing User Viewpoints in Requirements Definition, In *Proceedings of Eighth Australasian Conference on Information Systems*, Adelaide, pp. 1-14.

Darke, P., Shanks, G. and Broadbent, M. (1998) Successfully completing case study research: combining rigour, relevance and pragmatism, *Information Systems Journal*, 8, (4) pp. 273-289.

Dawson, L. (1997) Active, Non-interventionary Research into Object-Oriented Requirements Engineering, Seminar, Swinburne University of Technology, Melbourne.

Dawson, L. and Swatman, P. (1999a) The use of Object-oriented Models in Requirements Engineering: a Field Study, In *Proceedings of Twentieth International Conference on Information Systems*, (Eds, P. De and J. I. DeGross) Charlotte, NC, pp. 260-273.

Dawson, L. L., Milton, S. K. and Keen, C. D. (1995) A Functional Specification System for Information Systems, Technical Report 221, Department of Computer Science, Monash University, Melbourne, Australia.

Dawson, L. L. and Swatman, P. A. (1997) Object-Oriented Requirements Engineering in Practice, In *Proceedings of Fifth European Conference on Information Systems*, Cork, Ireland, pp. 1103-1112.

Dawson, L. L. and Swatman, P. A. (1998) The role of object-oriented modelling methods in requirements engineering, In *Proceedings of BCS Information Systems Methodologies*, (Ed. N. Jayaratna) University of Salford, Salford, UK, pp. 353-368.

Dawson, L. L. and Swatman, P. A. (1999b) The Role of Object-Oriented Modelling Methods in Requirements Engineering, In *Methodologies for Developing and Managing Emerging Technology Based Information Systems* (Eds, A. T. Wood-Harper, N. Jayaratna and J. R. G. Woods) Springer-Verlag, London, pp. 353-368.

DeMarco, T. (1978) *Structured Analysis and System Specification*, Yourdon Press, New York.

DeMarco, T. (1982) *Controlling Software Projects: Management, Measurement and Estimation*, Yourdon Press, New York.

Dey, I. (1993) *Qualitative Data Analysis: A User-Friendly Guide for Social Scientists*, Routledge, London.

Dorfman, M. and Thayer, R. H. (1990) *Standards, Guidelines and Examples on System and Software Requirements Engineering*, IEEE Computer Society Press, Los Alamitos, CA.

Downs, E., Clare, P. and Coe, I. (1988) *Structured Systems Analysis and Design Method: Application and Context*, Prentice Hall, Hemel Hempstead.

Dresner, D. C. (1996) Can anyone else use it?, World Wide Web, Accessed: Dec, 1996, http://www.avnet.co.uk/SQM/QiC/articles/Dresner/15/html.

Dubin, R. (1976) Theory Building in Applied Areas, In *Handbook of Industrial and Organisational Psychology* (Ed. M. Dunnette) Rand McNally College Publications, Chicago, pp. 17-39.

Duke, R., King, P., Rose, G. and Smith, G. (1991) The Object-Z specification language: Version 1, Technical Report, Software Verification Research Centre, Department of Computer Science, University of Queensland, Australia, Brisbane.

Duncker, K. (1945) On problem solving, *Psychological Monographs*, 58, (5).

Easterbrook, S. (1993) Domain Modelling with Hierarchies of Alternative Viewpoints, In *Proceedings of IEEE International Symposium on Requirements Engineering*, San Diego.

Eisenhardt, K. M. (1989) Building Theories from Case Study Research, *Academy of Management Review*, 14, (4) pp. 532-550.

Ericsson, K. A. and Simon, H. A. (1980) Verbal Reports as Data, *Psychological Review*, 87, (3) pp. 215-251.

Fichman, R. G. and Kemerer, C. F. (1992) Object-Oriented and Conventional Analysis and Design Methodologies, *IEEE Computer*, 25, (10) pp. 22-39.

Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L. and Goedicke, M. (1992) Viewpoints: A Framework for Integrating Multiple Perspectives in System Development, *International Journal of Software and Knowledge Engineering*, 2, pp. 31-57.

Fitzgerald, B. (1997) The use of systems development methodologies in practice: a field study, *Information Systems Journal*, 7, pp. 201-212.

Fitzgerald, B. (1998) An Empirically-Grounded Framework for the Information Systems Development Process, In *Proceedings of Nineteenth International Conference on Information Systems*, (Eds, R. Hirschheim, M. Newman and J. I. DeGross) Helsinki, Finland, pp. 103-114.

Fitzgerald, B. and Howcroft, D. (1998) Competing Dichotomies in IS Research and Possible Strategies for Resolution, In *Proceedings of Nineteenth International Conference on Information Systems*, (Eds, R. Hirschheim, M. Newman and J. I. DeGross) Helsinki, Finland, pp. 155-164.

Fitzgerald, G., Hirschheim, R. A., Mumford, E. and Wood-Harper, A. T. (1985) Information Systems Research methodology: An Introduction to the Debate, In *Research Methods in Information Systems* (Eds, E. Mumford, R. A. Hirschheim, G. Fitzgerald and A. T. Wood-Harper) North-Holland, Amsterdam, pp. 3-9.

Flynn, D. J. and Warhurst, R. (1994) An empirical study of the validation process within requirements determination, *Information Systems Journal*, 4, pp. 185-212.

Gadamer, H.-G. (1976) *Philosophical Hermeneutics*, University of California Press, Berkeley, CA.

Galal, G. (1998) Software Architecting:from requirements to building blocks within an architectural style, In *Proceedings of 12th European Conference on Object-Oriented Programming: ECOOP'98*, (Eds, I. Borne, M. Prieto, F. Brito e Abreu and W. De Meuter) Brussels.

Galal, G. and McDonnell, J. T. (1998) A Qualitative View of Requirements Engineering, In *Proceedings of Third Australian Conference on Requirements Engineering*, (Eds, D. Fowler and L. Dawson) Geelong, Australia, pp. 167-176.

Galliers, R. (1992) Choosing information systems research approaches, In *Information Systems Research: Issues, Methods and Practical Guidelines* (Ed. R. Galliers) Blackwell Scientific, Oxford, pp. 144-162.

Galliers, R. D. (1985) In Search of a Paradigm for Information Systems Research, In *Research Methods in Information Systems* (Eds, E. Mumford, R. A. Hirschheim, G. Fitzgerald and A. T. Wood-Harper) North-Holland, Amsterdam, pp. 281-298.

Galliers, R. D. and Land, F. F. (1987) Choosing appropriate information systems research methodologies, *Communications of the ACM*, 30, (11) pp. 900-902.

Gane, C. and Sarson, T. (1978) *Structured Systems Analysis: Tools and Techniques*, Prentice Hall, Englewood Cliffs, NJ.

Gardner, H. (1985) *The mind's new science:A history of the cognitive revolution*, Basic Books, New York.

Gentner, D. (1983) Structure mapping: A theoretical framework, *Cognitive Science*, 7, pp. 155-170.

Gentner, D. (1989) The mechanisms of analogical learning, In *Similarity and analogical reasoning* (Eds, S. Vosniadou and A. Ortony) Cambridge University Press, Cambridge, England.

Ghezzi, C., Jazayeri, M. and Mandrioli, D. (1991) *Fundamentals of Software Engineering*, Prentice-Hall, Englewood Cliffs, NJ.

Giarratano, J. and Riley, G. (1989) *Expert Systems: Principles and Programming*, PWS-Kent, Boston.

Gick, M. L. and Holyoak, K. J. (1980) Analogical problem solving, *Cognitive Psychology*, 12, pp. 306-355.

Glaser, B. (1992) *Basics of Grounded Theory Analysis*, Sociology Press, Mill Valley, CA.

Graham, I. (1994) *Object Oriented Methods*, Addison-Wesley, Wokingham, UK.

Graham, I., Henderson-Sellers, B. and Younessi, H. (1997) *The OPEN process Specification*, Addison Wesley, New York.

Greeno, J. G. and Simon, H. A. (1988) Problem Solving and reasoning, In *Stevens' handbook of experimental psychology* (Eds, R. C. Atkinson, R. J. Hernstein, G. Lindzey and R. D. Luce) Wiley, New York.

Greenspan, S., Mylopoulos, J. and Borgida, A. (1994) On Formal Requirements Modelling Languages: RM Revisited, In *Proceedings of 16th International Conference on Software Engineering*, Sorrento.

Guindon, R. (1990) Knowledge exploited by experts during software system design, *International Journal of Man-Machine Studies*, 33, pp. 279-304.

Hamilton, J. A. and Pooch, U. W. (1995) A survey of object-oriented methodologies, *Journal of the Association of Computing Machinery*, pp. 226-234.

Hamilton, S. and Ives, B. (1982) MIS Research Strategies, *Information and Management*, 5, pp. 339-347.

Harel, D. (1987) Statecharts:a Visual Formation for Complex Systems, *Science of Computer Programming*, 8, pp. 231-274.

Hawryszkiewycz, I. T. (1994) *Introduction to Systems Analysis and Design*, Prentice Hall, Englewood Cliffs, NJ.

Haywood, E. and Dart, P. (1999) Analysing projects to decide how to model the requirements, In *Proceedings of Fourth Australasian Conference on Requirements Engineering*, Sydney, Australia, pp. 149-159.

Henderson-Sellers, B. (1997) *A Book of Object-Oriented Knowledge*, Prentice-Hall, Upper Saddle River, NJ.

Henderson-Sellers, B. and Edwards, J. (1994) *BOOKTWO of Object-Oriented Knowledge: The Working Object*, Prentice Hall, Englewood Cliffs, NJ.

Henderson-Sellers, B. and Simons, A. J. H. (2000) OPEN-a third generation object-oriented methodology, *Journal of Research and Practice in Information Technology*, 32, (1) pp. 47-68.

Henderson-Sellers, B., Simons, A. J. H. and Younessi, H. (1998) *The OPEN Toolbox of Techniques*, Addison-Wesley Longman, Wokingham.

Hirschheim, R. and Klein, H. K. (1992) A Research Agenda for Future Information Systems Development Methodologies, In *Challenges and Strategies for Research*

in Systems Development (Eds, W. W. Cotterman and J. A. Senn) John Wiley and Sons, Chichester, pp. 235-255.

Hirschheim, R. A. (1985) Information Systems Epistemology: An Historical Perspective, In *Information Systems Research: Issues, Methods and Practical Guidelines* (Ed. R. D. Galliers) Blackwell Scientific Publications, Oxford, pp. 28-60.

Hirschheim, R. A. and Klein, H. K. (1989) Four paradigms of Information Systems Development, *Communications of the ACM*, 32, pp. 1199-1216.

Holyoak, K. J. and Koh, K. (1987) Surface and Structural Similarity in Analogical Transfer, *Memory and Cognition*, 15, pp. 337-340.

Hult, M. and Lennung, S.-A. (1980) Towards a definition of action research: a note and bibliography, *Journal of Management Studies*, 17, (May) pp. 241-250.

Humphrey, G. (1963) *Thinking: An introduction to its experimental psychology*, Wiley, New York.

IEEE-Std. (1990) IEEE Standard Glossary of Software Engineering Terminology, Institute of Electrical and Electronic Engineers, New York.

Jackson, M. C. (1997) Critical Systems Thinking and Information Systems Development, In *Proceedings of Eighth Australasian Conference on Information Systems*, (Ed. D. J. Sutton) University of South Australia, Adelaide, South Australia, pp. 1-20.

Jackson, P. (1990) *Introduction to Expert Systems*, Addison-Wesley, Wokingham, UK.

Jacobson, I. (1995) The Use-Case Construct in Object-Oriented Software Engineering, In *Scenario-Based Design: Envisioning Work and Technology in System Development* (Ed. J. M. Carroll) John Wiley and Sons, Inc, New York, pp. 309-336.

Jacobson, I., Booch, G. and Rumbaugh, J. (1999) *The Unified Software Development Process*, Addison Wesley Longman, Reading, MA.

Jacobson, I. and Christerson, M. (1995) Modeling with use cases: A growing concensus on use cases, *Journal of Object-Oriented Programming*, 8, (1) pp. 15-19.

Jacobson, I., Christerson, M., Jonsson, P. and Overgaard, G. (1992) *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley/ACM, New York.

Jarke, M., Bubenko, J., Rolland, C., Sutcliffe, A. and Vassiliou, Y. (1993) Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis, In *Proceedings of IEEE International Symposium on Requirements Engineering*, San Diego.

Johnson-Laird, P. N. (1983) *Mental Models*, Cambridge University Press, Cambridge.

Johnston, L. (1999) The Requirements Engineer as Architect?, In *Proceedings of Fourth Australasian Conference on Requirements Engineering*, Sydney, Australia.

Kaplan, B. and Duchon, D. (1988) Combining Qualitative and Quantitative Methods in Information Systems Research: A Case Study, *MIS Quarterly*, 12, (4) pp. 571-586.

Kaplan, B. and Maxwell, J. A. (1994) Qualitative Research Methods for Evaluating Computer Information Systems, In *Evaluating Health Care Information Systems: Methods and Applications* (Eds, J. G. Anderson, C. E. Aydin and S. J. Jay) Sage, Thousand Oaks, CA, pp. 45-68.

Keen, C. D., Lockwood, C. and Lamp, J. (1998) A Client-focussed, Team-of-Teams Approach to Software Development Projects, In *Proceedings of Software Engineering: Education and Practice*, IEEE Computer Society Press, Dunedin, New Zealand, pp. 34-41.

Keen, P. G. W. (1987) MIS Research: current status, trends and needs, In *Information Systems Education: Recommendations and Implementation* (Eds, R. A. Buckingham, R. A. Hirschheim, F. F. Land and C. J. Tully) Cambridge University Press, Cambridge, pp. 1-15.

Keen, P. G. W. (1991) Relevance and Rigor in Information Systems Research: Improving Quality, Confidence, Cohesion and Impact, In *Information Systems Research: Contemporary Approaches and Emergent Traditions* (Eds, H.-E. Nissen, H. K. Klein and R. Hirschheim) North-Holland, Amsterdam, pp. 27-49.

Khushalani, A., Smith, R. and Howard, S. (1994) What happens when designers don't play by the rules: Towards a model of opportunistic behaviour and design, *Australian Journal of Information Systems*, 1, (2) pp. 2-31.

Klahr, D. and Dunbar, K. (1988) Dual space search during scientific reasoning, *Cognitive Science*, 12, pp. 1-48.

Klein, H. K. and Myers, M. (1999) A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems, *MIS Quarterly*, 23, (1) pp. 67-93.

Kobryn, C. (1999) UML 2001: A Standardization Odyssey, *Communications of the ACM*, 42, (10) pp. 29-37.

Kotonya, G. and Sommerville, I. (1998) *Requirements Engineering: processes and techniques*, John Wiley and Sons, Chichester, UK.

Lamp, J., Keen, C. D. and Urquhart, C. (1996) Integrating Professional Skills into the Curriculum, In *Proceedings of First Australasian Conference on Computer Science Education*. Sydney, Australia, pp. 309-316.

Lamp, J. and Lockwood, C. (2000) Creating Realistic Experience of an IS Project: The Team of Teams Approach, In *Proceedings of 2000 IRMA International Conference*, Idea Group Publishing, USA, Alaska, pp. 737-740.

Lee, A. (1989) A Scientific Methodology for MIS Case Studies, *MIS Quarterly*, 13, (1) pp. 33-52.

Lee, Y. and Pennington, N. (1994) The effects of paradigm on cognitive activities in design, *International Journal of Human-Computer Studies*, 40, pp. 577-601.

Lockwood, C. and Lamp, J. (2000) Enhancing interpersonal skills in Information Technology Projects, In *Proceedings of Fourth World Multiconference on Systemics, Cybernetics and Informatics*, International Institute of Informatics and Systemics, USA, pp. 164-167.

Loosley, C., Mimo, A., Richards, D. and Winsberg, P. (1994) A Survey of Object-Oriented Methods, *InfoDB*, 9, (1) pp. 31-36.

Loucopoulos, P. and Karakostas, V. (1995) *Systems Requirements Engineering*, McGraw-Hill, London, UK.

Luchins, A. S. (1942) Mechanization in problem solving, *Psychological Monographs*, 54, (6).

Lyytinen, K. and Hirschheim, R. (1987) Information System Failures - A Survey and Classification of the Empirical Literature, *Oxford Surveys in Information Technology*, 4, pp. 257-309.

Macaulay, L. (1996) *Requirements Engineering*, Springer-Verlag, London.

Martin, J. and Odell, J. (1992) *Object-Oriented Analysis and Design*, Prentice-Hall, Englewood Cliffs, NJ.

Mayer, R. E. (1992) *Thinking, Problem Solving, Cognition*, W.H. Freeman and Company, New York.

Mayer, R. E. and Gallini, J. (1990) When is an illustration worth a thousand words?, *Journal of Educational Psychology*, 82, pp. 715-726.

Menzies, T. and Compton, P. (1995) The (Extensive) Implications of Evaluation on the Development of Knowledge-Based Systems, In *Proceedings of 9th AAAI-Sponsored Banff Knowledge Acquisition of Knowledge-based systems*, Banff.

Meyer, B. (1988) *Object-Oriented Software Construction*, Prentice-Hall, Englewood Cliffs, NJ.

Miles, M. B. and Huberman, A. M. (1994) *Qualitative Data Analysis: An Expanded Sourcebook*, Sage Publications Inc, Thousand Oaks, CA.

Monarchi, D. E. and Puhr, G. I. (1992) A Research Typology for Object-Oriented Analysis and Design, *Communications of the ACM*, 35, (9) pp. 35-47.

Morris, M., Speier, C. and Hoffer, J. (1996) The impact of experience on individual performance and workload differences using object-oriented and process-oriented systems analysis techniques, In *Proceedings of 29th Hawaii International Conference on System Sciences*, Maui, Hawaii.

Mumford, E., Hirschheim, R., Fitzgerald, G. and Wood-Harper, T. (1985) *Research Methods in Information Systems*, North Holland, Amsterdam.

MUSiC (1993) Metrics for usability standards in computing, World Wide Web, Accessed: October, 1996, http://newcastle.cabernet.esprit.ec.org/esp-syn/text/5429.html.

Myers, M. (1999) Qualitative Research in Information Systems, World Wide Web, Accessed: September, 10, 1999, http://www.auckland.ac.nz/msis/isworld/.

Nerson, J.-M. (1992) Applying Object-Oriented Analysis and Design, *Communications of the ACM*, 35, (9) pp. 63-74.

Neuman, W. L. (1994) *Social Research Methods: Qualitative and Quantitative Approaches*, Allyn and Bacon, Boston.

Norman, D. A. (1983) Some observations on mental models, In *Mental Models* (Eds, D. Gentner and A. L. Stevens) Lawrence Erlbaum Associates, Hillsdales, NJ, pp. 7-14.

Norman, D. A. (1988) *The Psychology of Everyday Things*, Basic Books, New York.

Nuseibeh, B., Kramer, J. and Finkelstein, A. (1994) A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification, *IEEE Transactions on Software Engineering*, 20, pp. 760-771.

OASIG (1996) Why do IT Projects so often Fail?", In *OR Newsletter*, Vol. 309, pp. 12-16.

Orlikowski, W. (1993) CASE tools are organisational change: Investigating Incremental and Radical Changes in Systems Development, *MIS Quarterly*, 17, (3) pp. 309-340.

Orlikowski, W. J. and Baroudi, J. J. (1991) Studying Information Technology in Organisations: Research Approaches and Assumptions, *Information Systems Research*, 2, pp. 1-28.

Pandit, M. R. (1996) The Creation of Theory: A Recent Application of the Grounded Theory Method, *The Qualitative Report*, 2, (4).

Pohl, K. (1993) The three dimensions of requirements engineering, In *Proceedings of Fifth International Conference on Advanced Information Systems Engineering*

(CAiSE'93), (Eds, C. Rolland, F. Bodart and C. Cauvet) Springer-Verlag, Paris, pp. 275-292.

Pohl, K. (1994) The Three Dimensions of Requirements Engineering: A framework and its applications, *Information Systems*, 19, (3) pp. 243-258.

Polya, G. (1957) *How to solve it*, Doubleday/Anchor, Garden City, New York.

Preece, J. (1994) *Human-Computer Interaction*, Addison-Wesley, Wokingham, UK.

Rapoport, R. N. (1970) Three Dilemmas in Action Research, *Human Relations*, 23, (4) pp. 499-513.

Reggia, J. (1985) Abductive Inference, In *Proceedings of Expert Systems in Government Symposium*, (Ed. K. Karna) IEEE Press, pp. 484-489.

Reitman, W. R. (1965) *Cognition and thought: An information processing approach*, Wiley, New York.

Robertson, J. and Robertson, S. (1997) *Volere Requirements Specification Template*, Atlantic Systems Guild, London, Aachen and New York.

Robertson, J. and Robertson, S. (2001) Volere Product Summary, World Wide Web, Accessed: Jan 23, 2001, http://www.atlsysguild.com/GuildSite/Robs/volprod.html.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W. (1991) *Object-Oriented Modeling and Design*, Prentice-Hall, Englewood Cliffs, NJ.

Schön, D. A. (1983) *The Reflective Practitioner: How Professionals Think in Action*, Basic Books, New York.

Shanks, G., Rouse, A. and Arnott, D. (1993) A Review of Approaches to Research and Scholarship in Information Systems, In *Proceedings of 4th Australasian Conference on Information Systems*, Brisbane.

Shlaer, S. and Mellor, S. J. (1988) *Object-Oriented Systems Analysis - Modelling the World in Data*, Yourdon Press, Englewood Cliffs, NJ.

Shlaer, S. and Mellor, S. J. (1991) *Modelling the World in States*, Yourdon Press, Englewood Cliffs, NJ.

Simons, A. (2000) Discovery, World Wide Web, Accessed: Feb, 2001, http://www.dcs.shef.ac.uk/~ajhs/discovery/discover.html.

Simons, A. J. H. (1998) Object Discovery - A process for developing medium-sized applications, In *Proceedings of ECOOP'98*, AITO/ACM, Brussels, pp. 109.

Simons, A. J. H. and Graham, I. (1998) 37 Things that don't work in object modelling with UML, In *Proceedings of 2nd. ECOOP Workshop on Precise Behavioural Semantics*, (Eds, H. Kilov and B. Rumpe) TU Munich, Brussels.

Simons, A. J. H. and Graham, I. (1999) 30 Things that go wrong in object modelling with UML 1.3, In *Behavioral Specifications of Businesses and Systems* (Ed. B. R. H Kilov, I Simmonds) Kluwer Academic Publishers, pp. 237-257.

Simons, A. J. H. and Swatman, P. (1997) Engineering the Object-Oriented Software Process: OPEN and MeNtOR, In *Proceedings of ECOOP'97 Tutorials*, AITO/ACM Press, Jyvaskyla.

Sommerville, I. (1996) *Software Engineering*, Addison-Wesley, Wokingham.

Sommerville, I. and Sawyer, P. (1997) *Requirements Engineering: A good practice guide*, John Wiley and Sons, Chichester.

Song, X. and Osterweil, L. J. (1994) Experience with an Approach to Comparing Software Design Methodologies, *IEEE Transactions on Software Engineering*, 20, (5) pp. 364-384.

Spivey, J. M. (1989) *The Z Notation: A Reference Manual*, Prentice Hall, Hemel Hempstead, UK.

Staudenmayer, H. (1975) Understanding conditional reasoning with meaningful propositions, In *Reasoning: Representation and process in children and adults* (Ed. R. J. Falmagne) Erlbaum, Hillsdale, NJ.

Staudenmayer, H. and Bourne, L. E. (1978) The nature of denied propositions in the conditional sentence reasoning task: Interpretation and learning, In *Human Reasoning* (Eds, R. Revlin and R. E. Mayer) Wiley/Winston, New York.

Strauss, A. and Corbin, J. (1990) *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*, Sage Publications, Newbury Park, CA.

Susman, G. I. (1983) Action Research. A Sociotechnical Systems Perspective, In *Beyond Method: Strategies for Social Research* (Ed. G. Morgan) Sage, Newbury Park, pp. 95-113.

Susman, G. I. and Evered, R. D. (1978) An Assessment of the Scientific Merits of Action Research, *Administrative Science Quarterly*, 23, (4) pp. 582-603.

Sutcliffe, A. G. and Maiden, N. A. M. (1992) Analysing the novice analyst: cognitive models in software engineering, *International Journal of Man-Machine Studies*, 36, pp. 719-740.

Swatman, P. A. (1996) Formal object-oriented method - FOOM, In *Specification of Behavioural Semantics in Object-Oriented Information Systems* (Ed. W. Harvey) Kluwer Academic Publishers, Norwell, Massachucetts.

Swatman, P. A. and Swatman, P. M. C. (1992) Formal Specification - an analytical tool for (management) information systems, *Journal of Information Systems*, 2, pp. 121-160.

Taplin, J. E. and Staudenmayer, H. (1973) Interpretation of abstract conditional sentences in deductive reasoning, *Journal of Verbal Learning and Verbal Behaviour*, 12, pp. 530-542.

Taylor, D. A. (1992) *Object-Oriented Information Systems*, John Wiley and Sons, New York.

The Concise Oxford Dictionary, (1976) Oxford University Press, Oxford.

The OPEN web page, (2000) URL, Accessed: 16/6/2000, 2000, http://www.open.org.au/.

Thorndike, E. L. (1898) Animal Intelligence: An experimental study of the associative processes in animals, *Psychological Monographs*, 2, (8).

Urquhart, C. (1998) Analysts and Clients in Conversation: Cases in Early Requirements Gathering, In *Proceedings of Nineteenth International Conference on Information Systems*, (Eds, R. Hirschheim, M. Newman and J. I. DeGross) Helsinki, Finland, pp. 115-127.

Vessey, I. and Conger, S. (1994) Requirements specification: Learning object, process, and data methodologies, *Communications of the ACM*, 37, (5).

Vidgen, R. and Braa, K. (1997) Balancing Interpretation and Intervention in Information Systems Research: The Action Case Approach, In *Proceedings of IFIP WG8.2 Working Conference on Information Systems and Qualitative Research*, (Ed. J. Liebenau) Philadelphia, USA.

Vitalari, N. P. and Dickson, G. W. (1983) Problem Solving for effective systems analysis: an experimental exploration, *Communications of the ACM*, 26, (11).

Walden, K. and Nerson, J.-M. (1995) *Seamless Object-Oriented Software Architecture. Analysis and Design of Reliable Systems.*, Prentice Hall, Hemel Hempstead, UK.

Walsham, G. (1995) Interpretive case studies in IS research: nature and method, *European Journal of Information Systems*, 4, pp. 74-81.

Wegner, P. (1987) Dimensions of object-based language design, In *Proceedings of OOPSLA*, ACM, New York.

Weidenhaupt, K., Pohl, K., Jarke, M. and Haumer, P. (1998) Scenarios in System Development: Current Practice, *IEEE Software*, 15, (2) pp. 34-45.

White, B. Y. and Frederoksen, J. R. (1987) Qualitative models and intelligent learning environments, In *Artificial Intelligence and education: Learning*

environments and tutoring systems (Eds, R. W. Lawler and M. Yazdani) Ablex, Norwood.

Winblad, A. L., Edwards, S. D. and King, D. R. (1990) *Object-Oriented Software*, Addison-Wesley, Reading, MA.

Wirfs-Brock, R. J., Wilkerson, B. and Wiener, L. (1990) *Designing Object-Oriented Software*, Prentice Hall, New York, USA.

Wynekoop, J. L. and Russo, N. L. (1997) Studying system development methodologies: an examination of research methods, *Information Systems Journal*, 7, pp. 47-65.

Yin, R. K. (1994) *Case Study Research: Design and Methods*, Sage Publications Inc., Thousand Oaks, CA.

Appendix A

The Categorised Interview Questions

Appendix A.1: The Initial Categorised Interview Questions

Category	Associated Interview Questions
THE PERSON	What is your name/ position/ job description
	How long have you worked for <organisation name>?
	How long have you been doing requirements engineering/systems analysis?
	Are you doing any object-oriented RE?
	How long have you been doing object-oriented RE?
	Have you done any formal course in RE/OO?
	What other methodologies have you used for system modelling and specification?
	How long have/did you used non-object-oriented modelling methods?
	Please comment on the advantages of the object-oriented methods (if any) over previously used methods.
THE PROJECT	What is the title of the project you are currently working on?
	Briefly describe the project.
	How was the project initiated?
	Who is the client(s)?
	What are the objectives of the project/system?
	How would you classify the project (eg real-time, transaction-based, etc?)
	What is the estimated time frame for this project?
	What platform/machine/OS will this project be running on?
	How many users will there be for this project?
	Are you the sole developer or are you part of a team?
	Do you have regular project meetings?
	Are you currently working on any other projects?
METHODOLOGY	Which method(ology) is being used?
	Why that method(ology)?
	Where does requirements specification fit within the methodology?
	Has this methodology been adopted across the organisation or only for this project?
DOCUMENTATION	What sort of documents do you start with (eg from the client)?
	Do you have any relevant in-house documentation that might help me understand your organisation and what it does?
	Do you use specific manuals, textbooks etc for reference?
THE RE PROCESS	
KNOWLEDGE ELICITATION	Is knowledge elicitation explicitly undertaken?
	When does knowledge elicitation begin?
Object-oriented elicitation	Is it seen as specifically object-oriented?
	What techniques and tools are used?
Feedback in elicitation	Is knowledge elicitation iterative?
	What is the time frame?

MODELLING	When does modelling begin? That is, when do you start drawing object models?
	What techniques and tools are used?
Which models	Which models are produced?
Static and dynamic models	Would you categorise models as static or dynamic?
Which models shown to users/development team	How are the models used?
	Who are they produced for?
	Which models, if any, are shown to the user?
VALIDATION	When does the validation process begin?
	What are the protocols? (In what order do you perform activities/tasks?)
	What techniques and tools are used?
	Who is involved in the validation process?
	When is the validation process considered to be complete?

Appendix A.2: The Final Categorised Interview Questions

Category	Associated Interview Questions
THE PERSON	
	What is your position/ job description in <organisation>
	How long have you worked for <organisation>?
	How long have you been doing requirements engineering/systems analysis?
	How long have you been doing object-oriented RE?
	Have you done any formal course in RE/OO?
	Have you used other (non-object-oriented) modelling methods for system modelling and specification?
THE PROJECT	
	What is the title of the project?
	Can you give me a brief description of the project?
	How was the project initiated? By tender?
	Who is the client(s)?
	What are the objectives of the project/system?
	How would you classify the project (eg real-time, transaction-based, etc?)
	What is the estimated time frame for this project?
	What platform/machine/OS will this project be running on?
	How many users will there be for this project?
	Are you the sole developer or are you part of a team?
	Do you have regular project meetings?
	Are you currently working on any other projects?
THE METHODOLOGY	
	Which method(ology) is being used on this project?
	Why that method(ology)?
	Where does requirements specification fit within the methodology?
	Has this methodology been adopted across the organisation or only for this project?
Methodology	Comment on the advantages of the current methodology (if any) over previously used methods.
Prototyping	Does prototyping play a large part in your system development method?
Implementation detail	Are you explicitly conscious of implementation details when you are doing requirements specification, or is the specification and modelling process completely implementation independent?
DOCUMENTATION	
	What sort of documents do you start with (eg from the client)?
	Do you have any relevant in-house documentation that might help me understand your organisation and what it does?
	Do you use specific manuals, textbooks etc for reference?
THE RE PROCESS	
KNOWLEDGE ELICITATION	
	Is knowledge elicitation explicitly undertaken and when does it start?
	Who is involved in the elicitation stage?
Thinking object-oriented from the beginning	Is it seen as specifically object-oriented, ie do you think "OO" from the start?
Mental models	Do you start developing mental models during elicitation?
Informal models	Do you use informal models (pictures) to communicate with the users?

Use case models	Do you use use case models at this stage? What form do they take?
	What techniques and tools are used? Interviews etc?
Feedback in elicitation	Is knowledge elicitation iterative? That is, do you go back to the users several times?
Opportunism in elicitation	Do you see the elicitation process as being sequential or does some piece of information trigger the need to explore some new feature or aspect?
	What is the usual time frame for the elicitation process?
MODELLING	
	When does modelling begin? That is, when do you start drawing object models?
	What techniques and tools are used?
Which models	Which models are produced during specification? Class models, use case models, interaction models, other?
Static and dynamic models	Would you categorise these models as static or dynamic?
Which models shown to users/development team	How are the models used?
	Who are they produced for?
	Which models, if any, are shown to the user?
	Which models are used internally by the development team?
Use case models	Could you comment on the role and/or importance of use cases in the specification process?
	Could you comment on the relationship between use cases and the more formal static and dynamic models?
Precedent and experience	How much of your requirements modelling technique, do you consider, comes from your knowledge and experience in other projects?
VALIDATION	
	Do you think it is necessary to validate the specification once the models have been produced?
	If so,
	How do you go about validating the specification or the models?
	When are you first prompted to start thinking about validation?
	When does the validation process start?
	What techniques and tools are used? Do you set up acceptance testing?
	How important are use cases in the validation process?
Prototypes in validation	How important are prototypes in the validation process?
	Who is involved in the validation process?
	When is the validation process considered to be complete?
RE as an ongoing process	Do you usually expect to revise or revisit the specification during design and implementation due to unforseen omissions, ambiguities etc?

Appendix B Published Papers Resulting from this Research

Dawson, L. (2001) "The use of Formal and Informal Models in Object-Oriented Requirements Engineering", in Proceedings of 3rd International Conference on Enterprise Information Systems, Setubal, Portugal, July 2001.

Dawson, L. (2001) "Template-Based Requirements Specification: A Case Study", in Proceedings of 3rd International Conference on Enterprise Information Systems, Setubal, Portugal, July 2001.

Dawson, L.L. and Silvas, A. (2000) "Requirements Specification for Electronic Service Delivery Applications Using 'Lazy Dog' Templates", in Proceedings of 2nd International Conference on Enterprise Information Systems, Stafford, UK, July 2000.

Dawson, L.L. and Swatman, P.A. (1999) "The use of object-oriented methods in requirements engineering: a field study", in Proceedings of 20th International Conference on Information Systems, Charlotte, USA, December 1999.

Dawson, L.L. and Swatman, P.A. (1999) "The role of object-oriented modelling methods in requirements engineering", in Methodologies for Developing and Managing Emerging Technology Based Information Systems, A.T. Wood-Harper, N. Jayaratna and J.R.G. Wood (Eds), Springer-Verlag, London, UK, 1999.

Dawson, L.L. and Swatman, P.A. (1997) "Object-oriented requirements engineering in practice", in Proceedings of 5th European Conference on Information Systems, Cork, Ireland, 1997.

Dawson, L.L. and Swatman, P.A. (1996) "Investigating the efficacy of object-oriented methods for requirements engineering", in Proceedings of First Australian Workshop for Requirements Engineering, Melbourne, September, 1996.