THE USE OF OBJECT-ORIENTED MODELS IN REQUIREMENTS ENGINEERING: A FIELD STUDY

Linda Dawson
Paul Swatman
School of Management Information Systems
Deakin University
Australia

Abstract

In many organizations, there has been a move toward the use of object-oriented methods for the development of information systems. Little is understood, or reported on the basis of research, of the use of object-oriented methods by practicing professionals in the production of requirements specifications for commercial or industrial sized projects. In this paper, we outline a conceptual framework of "what might be happening" in professional object-oriented requirements engineering based on the common characteristics of published, well known object-oriented methods. We then describe a research project and the findings from a set of six case studies that have been undertaken that examine professional practice from the standpoint of the epistemology contained in the conceptual model. In these studies, it was found that the more formal models of object-orientation were rarely used to validate, or even clarify, the specification with clients or users. Rather, analysts tended to use informal models, such as use cases or ad hoc diagrams, to communicate the specification to users. Formal models are more often used internally within the analysis team and for communicating the specification to the design team.

1. INTRODUCTION

The move toward 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 modeling (Graham 1994). Most of the commonly used texts emphasize object-oriented design (Booch 1991; Meyer 1988; Rumbaugh et al. 1991). 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, Henderson-Sellers and Younessi 1997; Jacobson, Booch and Rumbaugh 1999).

In order to understand what successful professional developers do, how they do it, and why they do what they do, further investigation is needed. This paper presents the foundations of a program of research into object-oriented requirements engineering. The case studies presented here indicate that analysts believe that users or clients cannot understand the more formal[1] models such as Object Modeling Technique (OMT) (Rumbaugh et al. 1991) or Unified Modeling Language (UML) (Booch,

[1] "Formal" in this context is not used in the same way as in mathematical or computer science literature. Rather, it is used to indicate models that use specific notation, e.g., OMT or UML diagrams.

Rumbaugh and Jacobson 1999) diagrams and interaction diagrams and that variations on use cases (Jacobson et al. 1992), ad hoc diagrams or rich pictures are a more appropriate informal models with which to communicate the specification to users.

Section 2 provides background in requirements engineering, object-oriented methods and related work. The research approach and an initial conceptual model reflecting object-oriented requirements engineering as described in the literature is presented in section 3. Section 4 describes six case studies of practicing professional requirements engineers. Section 5 presents the findings and interpretation of those findings. A discussion of the findings and a revised conceptual model based on the findings is presented in section 6 together with plans for ongoing research.

2. BACKGROUND

2.1 Requirements Engineering

Software engineering developed as a discipline to address the need for developing large, complex software systems in an organized and structured manner. Requirements engineering (Loucopoulos and Karakostas 1995; Macaulay 1996) specifically addresses the problems associated with errors in functional specifications that result in the costly maintenance or failure of software systems (Boehm 1976; Jackson 1997; OASIG 1996). Requirements engineering has now expanded to include more than just software construction (Loucopoulos and Karakostas 1995; Macaulay 1996) and is now an accepted term in information systems for defining the process and outcome of systems analysis.

Various definitions of, and approaches to, the requirements engineering process are suggested in the literature. Loucopoulos and Karakostas suggest a useful framework. In this framework, the requirements engineering process can be broken down into three subprocesses, elicitation, specification and validation, which deal with two external entities, the user and the problem domain. Kotonya and Sommerville (1998, p. 9) suggest that "there is no single [requirements engineering] process which is right for all organizations. Each organization must develop its own process which is appropriate for the type of systems it develops, its organizational culture, and the level of experience and ability of the people involved in requirements engineering."

Macaulay similarly suggests nine different approaches that "vary depending on the interest of the groups from which they originate." Pohl (1993) proposes three dimensions of requirements engineering. These three dimensions can be outlined as follows:

> *Representational* issues involve the range of representation methods from informal, natural language specifications to formal specification.
>
> *Social domain* issues encompass the social process involved in the iterative, co-operative elicitation of information for building requirements. Of particular importance is the aspect of communication between end users and analysts in establishing and documenting system needs within the organizational context.
>
> *Cognitive domain* issues are concerned with the understanding of the problem itself within the problem domain using various conceptual models.

The process model we present in section 3.1 is based on three subprocesses of elicitation, modeling and validation with specific notation for social and cognitive interactions.

2.2 Object-oriented Methods

Many claims have been made about the object-oriented paradigm (Booch 1994; Coad and Yourdon 1991; Henderson-Sellers and Edwards 1994; Meyer 1988). These claims include:

- ease of understanding object-oriented models due to a consistent underlying representation throughout the development process
- the ability to model the behavior of objects (encapsulation of data and process)
- ease of modification and extensibility of object-oriented models.
- ease of reuse of object components from previously designed systems
- superior data abstraction facilities including inheritance and polymorphism

When we study object-oriented methods in requirements engineering, our interest is in how experienced analysts and developers actually use such methods in "real-world" system specification. We also need to consider whether the benefits of reuse, abstraction and reduction in complexity outweigh any difficulties in using 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 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 phases.

Further, it is generally agreed within the object-oriented world 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 modeled as interactions between objects.

In this program of research, we open these key assertions to question and seek persuasive empirical evidence (for or against) that is grounded in the professional use of object-orientated methods on a commercial scale and in a commercial setting.

In an object-oriented modeling process, several models are usually produced. These models can be loosely categorized 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, Wilkerson and Wiener 1990), object/class models (Henderson-Sellers 1997), object and layer models (Graham 1994) and structural models (Swatman 1996).

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), object charts (Henderson-Sellers 1997), interaction diagrams (Jacobson et al. 1992), rules (Graham 1994), object communication models (Shlaer and Mellor 1991; Swatman 1996) and dynamic models using event chains and collaborative diagrams (Fowler 1996; Fowler, Swatman and Wafula 1995; Wafula and Swatman 1996). Sequencing is often modeled 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.

2.3 Related Work

Although little is understood about how practicing professionals use object-oriented methods and models for requirements determination and few empirical studies have been published that investigate this issue, 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 modeling (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 1995; Jacobson et al. 1992) as a "first system model …[which must] … be comprehensible by people both inside the development organization … and outside…. Object models are too complex for this purpose," but also suggests that use cases serve many other purposes as well (Jacobson and Christerson 1995). Some recent work has looked at 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 similar to those found in this study. That is, narrative text, structured

text, diagrammatic notations, images, animations and simulations. Graham (1994) suggests the use of task scripts that are "roughly similar to use cases."

There have been some studies of small groups, usually students in laboratory-type situations, where the participants use object-oriented methods for small one-off exercises and problems (Boehm-Davis and Ross 1992; Chaiyasut and Shanks 1994; Guindon 1990; Lee and Pennington 1994; Morris, Speier and Hoffer 1996; Sutcliffe and Maiden 1992; Vessey and Conger 1994). But the behavior of students, even graduate students, cannot be considered to be indicative of the behavior of experienced professionals without clear justification and/or demonstration. Also, the problems used in these experiments tend to be small and, therefore, not representative of commercial or industrial practice.

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, Speier and Hoffer 1996; Vessey and Conger 1994) have addressed specific aspects of using and learning object-oriented methods compared to other methods.

3. RESEARCH APPROACH

This research project investigates how object-oriented modeling methods are being used by practicing professionals for requirements engineering. The field research takes the form of a set of several case study interviews with individual professional consultants or systems analysts who are developing or have developed requirements specifications using object-oriented methods. The field interviews are based on an initial conceptual model that we have developed to reflect object-oriented requirements engineering as described in the literature. The case studies are used to refine and develop this initial conceptual model to a revised model that reflects object-oriented requirements engineering in practice.

The research approach used sequential, multiple case studies, which involved taped structured interviews with individual practicing professional requirements engineers within a specific research domain. The research domain was set by the conceptual model, which is grounded in the literature. Although a core set of interview questions remained the same for each case study (especially contextual/quantitative questions), each case sought to refine the conceptual 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

Reflection on, and re-examination of, data collected between cases led 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 model. The process is ongoing but can be 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). So, the outcome of the research method is a revised conceptual model (there have been several revisions during the process), which represents a theory about the area being investigated that is initially grounded in the literature and then progressively grounded in data gained from investigating the application of system development methods in practice.

3.1 An Initial Conceptual Model

Our initial conceptual model (Figure 1), revised from Dawson and Swatman (1999), is adapted from Loucopoulos and Karakostas (1995) and also contains elements based on the common characteristics of published, well known object-oriented methods (Avison and Fitzgerald 1995; Booch 1994; Coad and Yourdon 1991; Graham 1994; Henderson-Sellers 1997; Jacobson et al. 1992; Rumbaugh et al. 1991), particularly the concept of separate static and dynamic models.

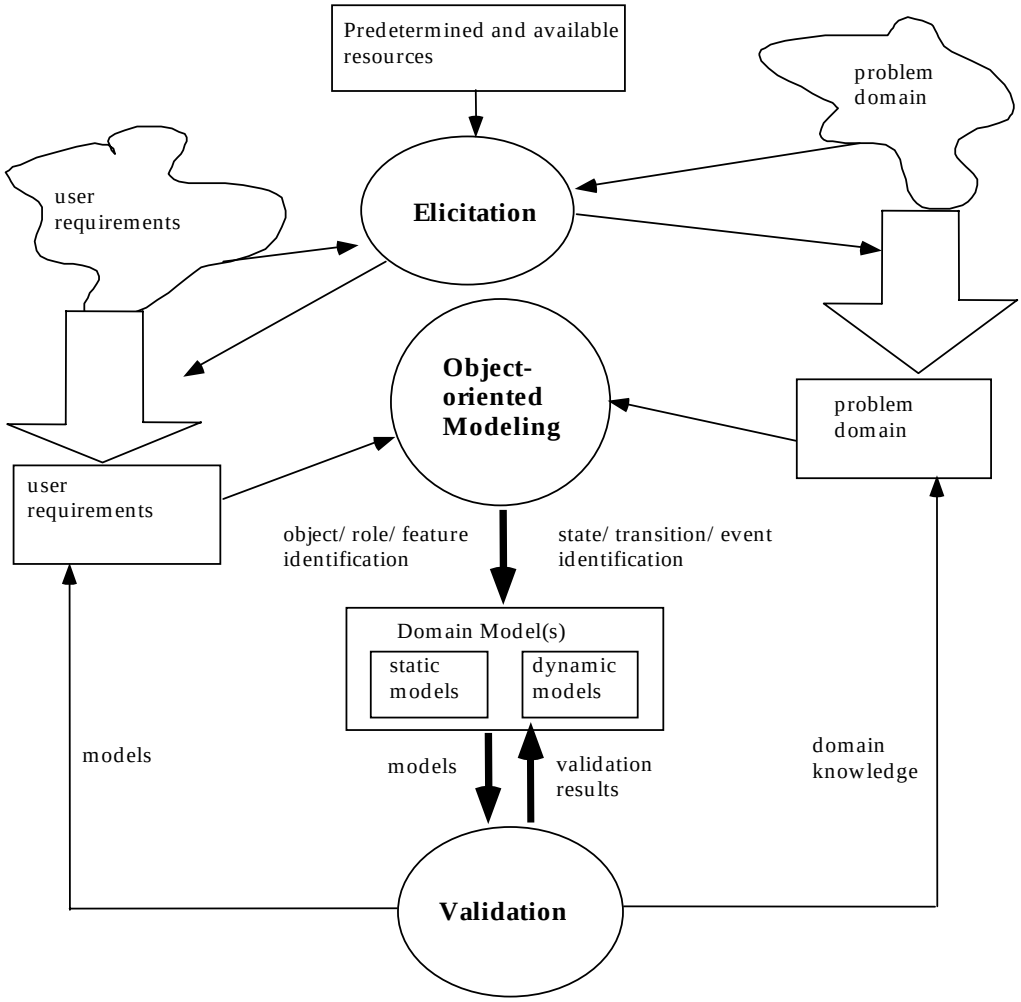

Figure 1. Conceptual Model

This model represents the three subprocesses of the requirements engineering process, elicitation, object-oriented modeling, and validation, which interact with users and a specific problem domain. The subprocesses are represented by circles.

3.1.1 The Elicitation Subprocess

This subprocess 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.

This elicitation process involves the transformation and clarification of user requirements and the problem domain until a view suitable for modeling is developed. This transformation is represented by the hollow arrows. One way to visualize this is to consider the elicitation process to be similar to the resolution of an image from low resolution containing few pixels to higher and higher resolution.

The thin arrows indicate interaction between the analyst and users or other actors, or knowledge and artefacts gained from users or other actors.

3.1.2 The Object-oriented Modeling Subprocess

The clarified knowledge (represented as rectangles) about the user requirements and the problem domain are then used as the basis of the object-oriented modeling subprocess. The initial model explicitly shows the static and dynamic models that are produced during object-oriented modeling. The thick arrows represent the flow of ideas and information within the cognitive domain of the analyst or analysts.

3.1.3 The Validation Subprocess

The validation subprocess takes these models and validates them against the user's original requirements based on the knowledge about the problem domain. The process of refining the models based on validation is based on feedback and is potentially unstructured.

4. THE CASE STUDIES

4.1 Unit of Analysis: The Analyst in Context

In order to understand the use of object-oriented models by practicing professionals, we must set the bounds of the research and data collection by determining the unit of analysis—the entity about which we want to gather data 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
- organizational change

The choice of the unit of analysis in a study is related to the way the questions and propositions are defined. In one sense, this project is concerned with all of the above categories but the main unit of analysis is the individual experienced requirements engineer or analyst, although within the context of four other elements derived from Yin's elements (see Figure 2).

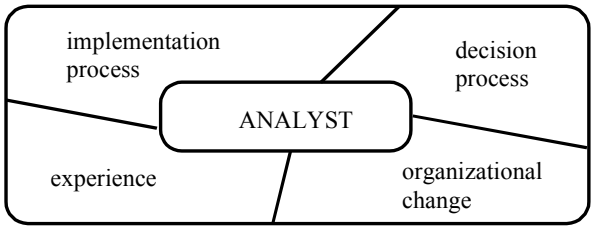

Figure 2. Unit of Analysis: The Analyst/ Requirements Engineer in Context

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

- *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 organization sponsored training course.

•

All of these aspects affect the gathering of data from the practicing professionals that participated in this study and these issues needed to be taken into account when formulating the interview questions.

4.2 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. 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 has been no attempt to select participants based on specific background characteristics. The common factor is that all of the participants were currently working in the field of object-oriented requirements specification. The contextual information for each consultant interviewed is summarized in Table 1.

Table 1. Background Information for Each Consultant

Case	Title	Years in RE	Years in OORE	Client	Project
1	Operations manager	3.5 yrs	0 yrs	Federal government	Complex Technical
2	Principal Consultant	15 yrs	10 yrs	State government	Web-based transactions
3	Senior Consultant	12 yrs	4 yrs	Telecommunications	Fault Management System
4	Director and 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

4.3 Data Collection Method

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 summarized in Table 2

Table 2. Types and Sources of Data and Their Collection Methods

Type of data	Source of data	Collection Method
Overall contextual data	Initial interview	Made by phone and followed up by e-mail 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 that emerged from the transcription process
Final clarification	Summary report	A report submitted to the participant with a final request for comment from participant

5. FINDINGS AND INTERPRETATION

Evaluating and analyzing rich qualitative data in a rigorous manner is difficult and "no single qualitative data analysis approach is widely accepted" (Neuman 1994). The approach in this project was to use an evolving set of categories to structure the

qualitative data as it was gathered. First, a set of seed categories (Fitzgerald 1997; Miles and Huberman 1984; Wynekoop and Russo 1997) (see Table 3) was formulated based on the initial conceptual model and these were used to formulate the initial structured interview script (see sample questions in column 2, Table 3). Subsequently, the transcript of each interview was partitioned and distilled into a template structure reflecting the current set of categories. In each interview, other categories and subcategories emerged and were incorporated into interview scripts and templates for investigation in the following cases, so the number of categories grew as the case studies continued. The final analysis for each of the findings outlined in this paper were based on the following chain of processes:

- **Revelation** of emergent new categories, or subcategories, 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 modeling by the analyst. This idea, or emergent category, of "mental modeling" was 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 modeling had occurred. Other issues raised in the interviews led to other new categories in a similar manner. These findings are outlined in Tables 4, 5, and 6.

Table 3. Research Issues and Example Questions

Seed categories	Example questions
Elicitation How is knowledge elicitation influenced by the use of object-oriented modeling methods?	Is knowledge elicitation explicitly undertaken? Is it seen as object-oriented? Is it seen as sequential or opportunistic?
Modeling When, how and for whom is object-oriented modeling undertaken?	When does modeling 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 4. Evidence of Use Case Modeling

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

5.1 Use Cases

In Case 2, the use of simplified use cases as models in requirements specification emerged as a significant technique. Case 1 used a methodology that potentially included use cases but the analyst did not use them. Subsequently, Case 3 and, to a lesser extent Cases 5 and 6, gave evidence of the use of use cases for requirements modeling. Table 4 summarizes the evidence from the emergence, reinforcement and re-examination for the use of use cases for requirements modeling.

Some illustrative quotes:

> *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 modeled 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.*

> *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.*

5.2 Mental Modeling

Four of the six analysts believed that they were continually "modeling 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.

Table 5. Evidence of Mental Modeling

Case No.	Use of mental modeling
Case 1	Didn't mention mental modeling
Case 2	Didn't mention mental modeling
Case 3	Mental modeling emerged as significant
Case 4	Mental modeling seen as a natural part of requirements modeling
Case 5	Mental modeling perceived specifically in terms of abstraction
Case 6	Mental modeling seen as natural to analyst personally (not necessarily for other professional analysts)

Some illustrative quotes:

> *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.*

> *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.*

5.3 Formal and Informal Models

As described above, when asked when the modeling process began, most analysts said they were building models in their heads long before any formal models were written down.

For the purpose of discussion, we distinguish between formal models and informal models 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. Evidence of the use of separate models is shown in Table 6.

Three of the cases used prototypes early in the specification process. One case used coded sections of the system that provided sample screens for discussion with the users. In another case, an animation package was used. The third case used a standard presentation package (PowerPoint) 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 modeling direct with clients. Some also used ad hoc diagrams and one used rich pictures (Checkland and Scholes 1990) to explain and communicate the specification to users.

In the case studies, there seemed to be these two different kinds of modeling taking place. First, there were the informal models that were used to communicate with the users. Second, 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 kept internal 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 the formal models were refined accordingly. When asked how this mapping process was done, most analysts had difficulty describing the process. It was described as being able to think in terms of either type of model depending who 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.

Table 6. Evidence of the Separation of Models

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 and interaction models
Case 3	Use case scripts & prototype	OMT class and 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, prototype and use cases for exceptions and special cases	OMT class & interaction models

Some illustrative quotes (researcher's questions are shown in bold type)

> **Did you show this group [of users] use cases?** *Yes.* **And what about the object models?** *We would have stopped short there.*

We tell them [the users] that the model is technical mumbo jumbo.

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.

In one case, 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*."

6. DISCUSSION

The findings from the case studies presented here have led to the revision of the initial conceptual model, Figure 3. This revised model is based on the following characteristics:

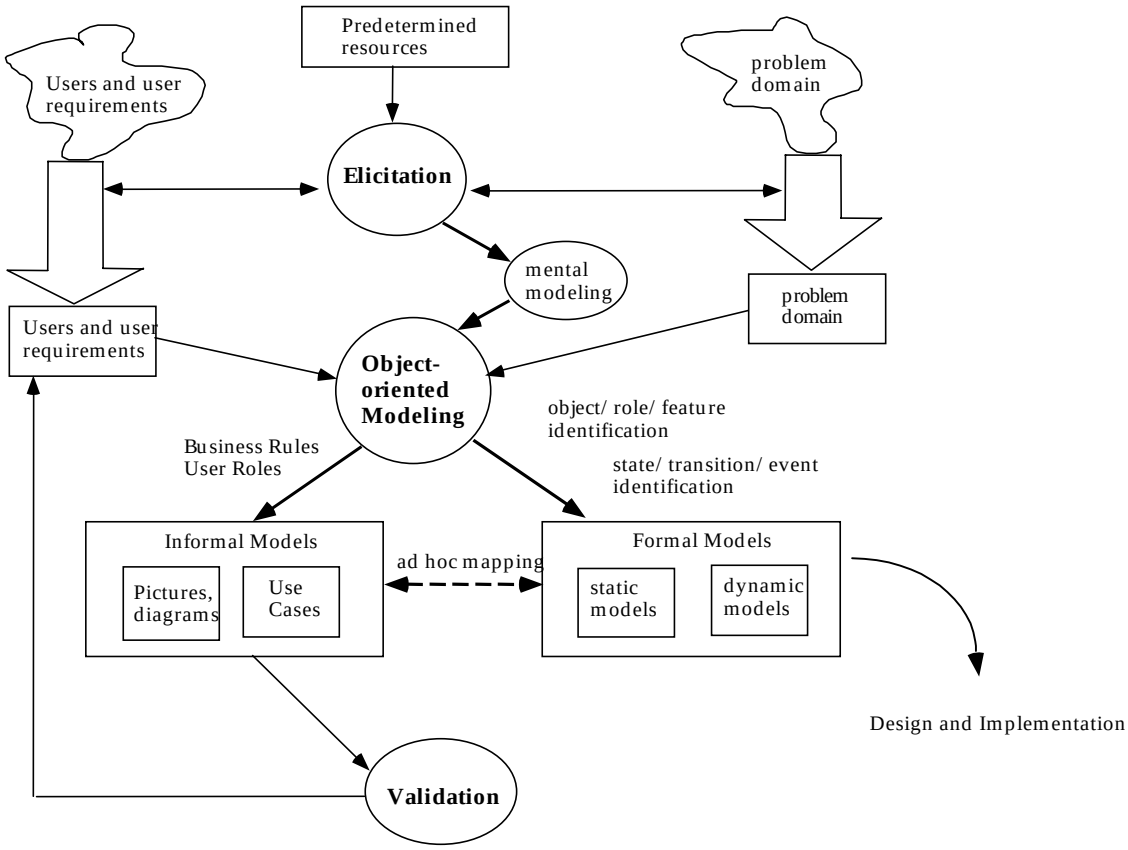

Figure 3. Revised Conceptual Model

- Four of the six case studies described a process of mental modeling during elicitation and this is now shown explicitly in the model.

- There was evidence that the modeling process produced two types of models: informal models in the form of pictures, text and diagrams and/or use cases and prototypes; and formal models in the form of class models, object models and to a lesser extent interaction and event models.

- 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.

- 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 subprocess was informal and formal models were not used in this subprocess.

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 basis for this belief may lie in the concept that the requirements engineering process is fundamentally a social process involving two main groups: the users/clients and the professional consultants. If we believe that 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 from both points of view—the *formal* or consultant's point of view and the client's *informal* point of view. The perception seems to be that for this agreement to take place, there needs to be two levels of modeling: informal modeling for communicating the specification to the user for information and validation; and formal modeling for the analyst team to pass on to the design and implementation team.

7. CONCLUSION

The implications for practice in these findings lie in the recognition of the social aspects of the requirements specification process (Macaulay 1996; Urquhart 1998). The errors that arise due to inconsistencies, omissions and ambiguities in functional specifications often result in the costly maintenance or failure of software systems (Boehm 1976; DeMarco 1982; Jackson 1997; OASIG 1996). If, as the findings of this research project suggest, the models used for validation of the specification with the clients are different from the models used in design and implementation, then this may indicate one of the areas where these inconsistencies, omissions and ambiguities might arise. The extension of use case models and concrete scenarios as suggested in Weidenhaupt et al. (1998) may assist in addressing these issues.

Further research, in the form of a survey of professional systems developers, is currently being investigated. This survey will be based on the field documents and interview scripts used in the case studies and will be designed to further test and improve the revised conceptual model produced from the findings of the case studies presented in this paper. Ultimately, we expect to gain more knowledge of how large real world organizational systems have been specified and developed and which tools and techniques facilitate the successful specification for the design and implementation of useable systems.

8. ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees whose constructive comments aided in the revision of this manuscript.

9. REFERENCES

Avison, D. E., and Fitzgerald, G. *Information Systems Development: Methodologies, Techniques and Tools,* London: McGraw Hill, 1995.

Boehm, B. W. "Software Engineering," *IEEE Transactions on Computers* (25), 1996, pp. 1226-1241.

Boehm, B. W. "A Spiral Model of Software Development and Enhancement," *IEEE Computer* (25), 1988, pp. 61-72.

Boehm-Davis, D., and Ross, L. "Program Design Methodologies and the Software Development Process," *International Journal of Man-Machine Studies* (36), 1992, pp. 1-19.

Booch, G. *Object-Oriented Design,* Menlo Park, CA: Benjamin/Cummings, 1991.

Booch, G. *Object-Oriented Analysis and Design with Applications,* Redwood City, CA: Benjamin/Cummings, 1994.

Booch, G.; Rumbaugh, J.; and Jacobson, I. *The Unified Modeling Language User Guide,* Reading, MA: Addison-Wesley, 1999.

Chaiyasut, P., and Shanks, G. "Conceptual Data Modeling Process: A Study of Novice and Expert Data Modelers," in *Proceedings of First International Conference on Object Role Modeling*, Magnetic Island, Australia, 1994.

Checkland, P., and Scholes, J. *Soft Systems Methodology in Practice,* Chichester, England: Wiley, 1990.

Coad, P., and Yourdon, E. *Object-Oriented Analysis,* Englewood Cliffs, NJ: Yourdon Press/Prentice-Hall, 1991.

Dawson, L. L., and Swatman, P. A. "The Role of Object-Oriented Modeling 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. Woods (eds.), London: Springer-Verlag, 1999, pp. 353-368.

DeMarco, T. *Controlling Software Projects: Management, Measurement and Estimation,* New York: Yourdon Press, 1982.

Eisenhardt, K. M. "Building Theories from Case Study Research," *Academy of Management Review* (14), 1989, pp. 532-550.

Fitzgerald, B. "The Use of Systems Development Methodologies in Practice: A Field Study," *Information Systems Journal* (7), 1997, pp. 201-212.

Fowler, D. *Formal Methods in a Commercial Information Systems Setting: The FOOM Methodology*, unpublished Ph.D. Thesis, Centre For Information Systems Research, Swinburne University of Technology, Melbourne, 1996.

Fowler, D.; Swatman, P. A.; and Wafula, E. N. "Formal Methods in the IS Domain: Introducing a Notation for Presenting Object-Z Specifications," *Object Oriented Systems* (2), 1995.

Graham, I. *Object Oriented Methods,* Reading, MA: Addison-Wesley, 1994.

Graham, I.; Henderson-Sellers, B.; and Younessi, H. *The OPEN Process Specification,* Reading, MA: Addison-Wesley, 1997.

Guindon, R. "Knowledge Exploited by Experts During Software System Design," *International Journal of Man-Machine Studies* (33), 1990, pp. 279-304.

Henderson-Sellers, B. *A Book of Object-Oriented Knowledge,* Upper Saddle River, NJ: Prentice-Hall, 1997.

Henderson-Sellers, B. and Edwards, J. *BOOKTWO of Object-Oriented Knowledge: The Working Object,* [illegible]

Morris, M.; Speier, C.; and Hoffer, J. "The Impact of Experience on Individual Performance and Workload Differences Using Object-oriented and Process-oriented Systems Analysis Techniques," in *Proceedings of Twenty-ninth Hawaii International Conference on System Sciences*, Maui, Hawaii, 1996.

Neuman, W. L. *Social Research Methods: Qualitative and Quantitative Approaches,* Boston: Allyn and Bacon, 1994.

OASIG. In *OR Newsletter* (309), 1996, pp. 12-16.

Pohl, K. "The Three Dimensions of Requirements Engineering," in *Proceedings of Fifth International Conference on Advanced Information Systems Engineering (CAiSE'93)*, Paris, 1993, pp. 275-292.

Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; and Lorensen, W. *Object-Oriented Modeling and Design,* Englewood Cliffs, NJ: Prentice-Hall,1991.

Shlaer, S., and Mellor, S. J. *Modeling the World in States,* Englewood Cliffs, NJ: Yourdon Press,1991.

Sutcliffe, A. G., and Maiden, N. A. M. "Analyzing the Novice Analyst: Cognitive Models in Software Engineering," *International Journal of Man-Machine Studies* (36), 1992, pp. 719-740.

Swatman, P. A. "Formal Object-oriented Method: FOOM," in *Specification of Behavioral Semantics in Object-Oriented Information Systems*, W. Harvey (ed.), Norwell, MA: Kluwer Academic Publishers, 1996.

Urquhart, C. "Analysts and Clients in Conversation: Cases in Early Requirements Gathering," in *Proceedings of Nineteenth International Conference on Information Systems*, R. Hirschheim, M. Newman, and J. I. DeGross (eds.), Helsinki, Finland, 1998, pp. 115-127.

Vessey, I., and Conger, S. "Requirements Specification: Learning Object, Process, and Data Methodologies," *Communications of the ACM* (37), 1994.

Vitalari, N. P., and Dickson, G. W. "Problem Solving for Effective Systems Analysis: An Experimental Exploration," *Communications of the ACM* (26), 1983.

Wafula, E. N., and Swatman, P. A. "FOOM: A Diagrammatic Illustration of Object-Z Specifications," *Object Oriented Systems* (3), 1996, pp. 215-243.

Weidenhaupt, K.; Pohl, K.; Jarke, M.; and Haumer, P. "Scenarios in System Development: Current Practice," *IEEE Software* (15), 1998, pp. 34-45.

Wirfs-Brock, R. J.; Wilkerson, B.; and Wiener, L. *Designing Object-Oriented Software,* Englewood Cliffs, NJ: Prentice Hall, 1990.

Wynekoop, J. L., and Russo, N. L. "Studying System Development Methodologies: An Examination of Research Methods," *Information Systems Journal* (7), 1997, pp. 47-65.

Yin, R. K. *Case Study Research: Design and Methods,* Thousand Oaks, CA: Sage Publications Inc.,1994.