OBJECT-ORIENTED ANALYSIS: A VOYAGE OF DISCOVERY

Elizabeth A. Kemp
Institute of Information Sciences and Technology
College of Sciences
Massey University
Palmerston North, New Zealand
Email: E.Kemp@massey.ac.nz

ABSTRACT

In this paper the experiences of a business analyst who worked on several projects using the object-oriented (O-O) approach are described. The business analyst in question had initially no background whatsoever in O-O and worked within organisations which were also making the same transition. Her reflections on the events that transpired are discussed and some suggestions made for organisations migrating to O-O.

INTRODUCTION

Employing the Object-Oriented (O-O) approach for application development has been acknowledged to have benefits for certain applications: higher quality software, more flexibility, code re-use, greater resilience to change etc (Graham, 1994; Booch 1996). Nonetheless, for individuals and organisations, the migration to this paradigm is non-trivial and involves a considerable effort (Meyer, 1995; Goldberg and Rubin, 1995; Booch, 1996). In this study, a business analyst was interviewed on three separate occasion over a fifteen month period about her experiences learning the object-oriented paradigm. The underlying objective was to find out how one individual working in the computing industry came to grips with the move to the object-oriented paradigm. The business analyst described her role in various projects and reflected on her experiences. Because she was working within businesses that were also engaged in the transition to O-O, it also proved possible to think about lessons to be learned for organisations when implementing the O-O approach. Two voices can be heard in this paper, that of the business analyst and that of the author. Initially the emphasis is on the business analyst with her account of the three projects on which she worked and what she felt that she had learned. The author then analyses the findings, taking account of the literature currently available to organisations migrating to O-O.
Research of the kind described here falls within the interpretivist method of research where knowledge is socially constructed. It is not possible to make any statistical generalisations from the results. Instead, qualitative research offers the opportunity to describe and understand what is happening in the world. As Walsham (1995, p 6) notes "*There are no correct and incorrect theories but there are interesting and less interesting ways to view the world.*" It is hoped that the research described here will make a contribution to the debate about the migration to object-oriented technology and will be more rather than less interesting.

METHODOLOGY

There are a variety of philosophical views in relation to the nature of reality. Positivists assume the existence of a single objective reality that can be observed and expressed in terms of universal laws and empirical generalisations (Keat, 1981). Others hold a subjectivist ontological position where they believe in multiple realities that exist as mental constructions whereby people interpret and make sense of the world (White et al., 1992). Human behaviour is continually constructed and reconstructed on the basis of individual's interpretation of the situations they are in (Hammersley and Atkinson, 1983). To access people's meanings, the researcher cannot be objective and external to the phenomena under investigation, rather the researcher must play the role of the participant observer (Hammersley and Atkinson, 1983). It is this role which is played by the author of this paper who both describes and comments on what occurred. The author has several years experience of both teaching and research in software engineering, focusing recently on the O-O paradigm.
An in-depth, longitudinal case study approach was followed. Walsham (1995) observes that this is appropriate in the interpretivist tradition. A longitudinal study allows for the unfolding of events over time to be studied. Benbasat et al. (1987) maintain that case research is particularly appropriate for practice-based problems where research and theory are still being developed. It can be argued that studies of O-O practitioners are still at that stage. Walsham, (1995, p 15), notes that any extrapolations from case studies of this kind rely on the "*plausibility and cogency of the logical reasoning used and in drawing conclusions from them.*" It is also possible, however, to compare the findings with the literature on making the transition to the object-oriented paradigm.

One business analyst was chosen as the subject of the case study. She was a suitable candidate for various reasons: her degree included papers in systems analysis and she had several years experience in the computing industry, specialising in data modelling. The business analyst, therefore, had a suitable background for someone learning the O-O approach. She was, moreover, keen to participate and would make herself available when required.

During a fifteen month period, the business analyst was interviewed on three separate occasions: about her experiences as a complete novice, when familiar with the main concepts of the O-O approach and after completing the analysis phase of a large project. The interviews were unstructured, allowing the business analyst to describe her own experiences and reflect on coming to grips with the O-O paradigm. The business analyst checked that the account of events outlined in this paper was correct. For reasons of confidentiality, details of examples have been changed when necessary.

ORGANISATIONAL CONCERNS

Advice on managing the transition to object-oriented software development is provided by various authors. Hares and Smart (1994) talk of crossing the "Rubicon" when the move is made to an object-oriented environment. They discuss what types of computer systems to migrate and when to make the transition. Confidence in the technology is an important prerequisite for making the move. Booch (1996) addresses the project management issues that may arise in his text 'Object Solutions." He believes that ideally an organisation should start with a pilot project since this allows important learning to take place in a controlled environment. Staff should be adequately trained and suitable tools made available to them. Even so, Booch warns that it might take several months for programmers to be at ease with the paradigm. Using experienced members of the team to mentor others may alleviate some of these problems. Booch also describes a macro and micro process that can be followed when developing O-O software. In particular he emphasises the need for a requirements phase during which an O-O model of the problem should be built. Involving real users throughout the project is seen as vital. Finally the project manager is warned that if more than two existing practices (such as the terminology, the platform, the method, the staff, the tools, the development language etc) are changed then the outcome is uncertain.

Cockburn has recently written a book called "Surviving Object-oriented Projects." He comments that organisations have claimed significant reduction in development time but that people *"rarely seem to want to name the actual costs of making the move to objects, perhaps to avoid scaring away future newcomers. There is, therefore, a lack of information on what sorts of unpleasant surprises await one when starting an O-O project"* (Cockburn, 1998, p xv). Rebecca Wirks-Brock has also commented that introducing objects may create more problems than it solves. (Cockburn, 1998, p xiii). Cockburn provides a wealth of advice to managers working on their first O-O project. A suitable project must be selected,*"not too big, not too small, not too important, not too unimportant"* (Cockburn, 1998, p 33), and an appropriate methodology selected. Careful planning is required with standards established; incremental and iterative procedures should be followed. He warns managers that they must invest in training their staff. Managers should not have unrealistic expectations but expect programmers converting to the O-O paradigm to take 6 to 9 months to become productive. Access to people experienced in the technology is also required. Both consultants and mentors are needed. Consultants can help an organisation avoid pitfalls whilst the role of the mentor is to help developers acquire good habits. As Cockburn points out, however, experience is a relative term in an immature industry. *"More object projects are starting up than there are experts to lead them, and even large consultancy houses are having difficulty finding experienced O-O developers...After you train novices for a year, they may quit and become self-declared experts* "(Cockburn, 1998, p 44). Consequently, managers are at risk of hiring experts with only one year's experience.

Ambler (1998) goes beyond organisational issues to discuss ways in which individuals can help themselves overcome the O-O learning curve. The following activities were seen as useful: reading books, magazines and journals on the topic, attending courses and conferences, browsing the internet and obtaining a mentor.

GETTING STARTED

With a degree in English and Information Systems, the first position held by the business analyst was in the area of data administration for a large organisation. She carried out a wide variety of tasks including data modelling and the migration of software modules into production from maintenance environments. Code tables also had to be kept up to date. She was a member of a team that built a corporate data model for the organisation. In addition she was also responsible for reviewing the standards and methodology manuals for software development.

At this point, with around three years of experience, a move was made to a position with a major consulting organisation. One of her first tasks was to produce the analysis for a small system to be written in Java. There was no time to become familiar with the object-oriented approach that would have been most appropriate in the circumstances. Instead the analysis was centred initially around a textual description of user requirements and the production of an Entity Relationship Diagram (ERD). The ERD formed the basis of the normalised, physical data model for a relational database. The programmer took the entities as the basis for classes and objects. Subsequently dataflow diagrams and an event list were drawn up. There was also the requirement to identify the associated methods. The dataflow diagrams proved to be of little value in this respect. Instead the methods were taken from the screen designs (derived from the event list) produced by the business analyst. The process of developing the class diagram and physical data model was more iterative than strictly necessary due to inexperience of the team. The system was quickly up and running, however, and proved its value to the organisation.

The business analyst recognised, though, that she was too dependent on the tools and techniques of structured analysis. When the analysis for a second (prototype) system was required, she researched various approaches to the object-oriented paradigm, reading several texts as well as material from the web and trying out the Rational Rose Case tool. She based her approach to this project on the method proposed in "Case Studies in Object-oriented Analysis and Design" by Yourdon and Argila (1996).

First the business analyst produced the written user requirements documentation with the aim of specifying ultimately the relevant business objects, attributes and methods. Then, following Yourdon and Argila's process of textual analysis, she identified the significant nouns, about 15 to 20. These represented the candidate objects. To recognise significant objects, the 3-view approach of building Entity Relationship Diagrams (ERD) etc was used. A large ERD diagram was drawn up before developing a class diagram. Entities, however, could not just be mapped across into objects. One difference, according to Yourdon and Argila is that an object does not have to hold data eg. "funds transfer." Nor do all associative entities have to be represented as objects. Object models can show many to many relationships. Nonetheless, the business analyst found it difficult as a novice analyst to move away from the relational approach. There was the requirement to produce a physical data model for the relational database which was based on the ERD as well as a class diagram. This caused her some problems. She initially moved across an attribute such as "account type" into a class diagram as an attribute of an object when the different types of account should have been represented by an "is-a" relationship. Her advice to others was to *"be aware when you are thinking relationally and be aware when you are thinking object-oriented."*

The attributes of the objects were taken directly from the logical/physical data models. They had been obtained initially from the user requirements specification and also research on the subject such as an analysis of forms from banks. Experience indicated that attributes should be checked with users but that they will never have a full list of attributes in their head.

The methods to be associated with objects were also identified by the business analyst, following a procedure suggested by Yourdon and Argila (1996). This involved going through the life cycle of an object imagining what it does from start to finish. The analyst also drew up an event list which assisted with this process. She believed, though, that the resulting class diagram was in some ways too detailed (all objects had create, update and delete associated with them) and in others incomplete. Whilst she identified all major methods it proved difficult to think of all the optional activities that might occur. Associating a method with the appropriate object also proved problematic, *"sometimes you are not sure where to put the methods."* In one case, there were three objects to which a particular method could be attached.

The event list, however, whilst it did not prove particularly useful for method identification and placement, was seen as invaluable for validation purposes. A check was made that every event was supported by the business object diagram. The business analyst also had to verify that the activities of the class diagram were supported by the physical data model. Only a few small omissions were made which were quickly rectified.

Overall, the business analyst was concerned about the problems caused by having to develop both ERDS and a class diagram. She asked herself whether Yourdon's recommended starting point (ERD views) was appropriate. Another member of the consultancy advised her to discard this approach and move to one that was more object-oriented. How though, the analyst wondered, would she be able to move from a textual user requirements documents to a class diagram and be sure that she was retaining everything of importance.

Even though the business analyst was at this point essentially a novice she was still able to stand back and comment on her experiences to this point. She quickly realised after the first project that she needed to familiarise herself further with O-O techniques and turned to Yourdon and Argila (1996). She had been taught during her university days Yourdon's (1989) approach to structured systems analysis and thought that it made sense to move in a systematic way from a structured approach to O-O. On reflection, however, she believed that this approach ultimately proved confusing, especially with regard to the identification of objects and methods After discussing with others in the consultancy the problems of moving from an ERD to a class model, she was

advised to start in future from a purer O-O approach. There was still the underlying problem, however, of underpinning the O-O approach by a relational database.

A PRACTICE RUN

By the time the business analyst started work, as a member of a team, on a large (two year) transaction processing project, she had, therefore, had experience of using one approach to O-O analysis. She, together with other analysts, had also attended a course on UML (Unified Modelling Language). A comprehensive O-O analysis methodology had been worked out for this project by the analysis group. A great deal of research had been undertaken which included searching the web for relevant information. The project involved the rewriting and upgrading of an existing system. The first stage involved a practice run of tools and techniques to show that the approach selected would work. This proof of concept for one subsystem (customer management) was expected to take six weeks from initiation to implementation (using Java and Corba2). For the organisation concerned, this was its first venture into O-O.

Some important techniques that can be used in analysis, taken from a guide to Unified Modelling Language (UML) by Booch, Rumbaugh and Jacobsen (1998), will be described here. One of these is the use case. A use case essentially is a description of a sequence of actions carried out by a system which provides a result to an actor. A use case diagram, therefore, is one which shows a set of use cases and actors. Actors in their turn are defined in terms of "*a coherent set of roles that users of use cases play when interacting with the use cases*" (Booch et al, 1998, p 457). Figure 1 shows a use case diagram for the situation where a system is being developed for a garage. The system, *inter-alia*, will handle the recording of repairs made to vehicles and the ordering of parts required. The actors ("manager" and "mechanic") are represented by stick figures and the use cases ("place order" and "record repair details") as ellipses. Any individual can play more than one role, for example the manager of the garage can also act on occasion as a mechanic.

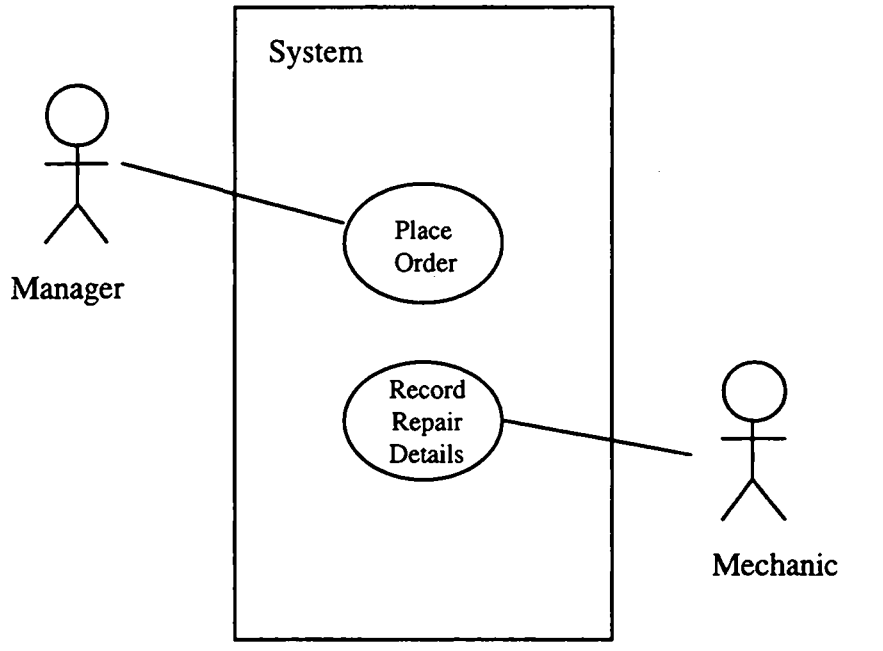

Figure 1: An example of a use case

Scenarios describe the flow of events for a use case. There may be primary and secondary scenarios. Essential sequences of actions are incorporated into primary scenarios and alternative sequences in secondary scenarios. To help identify the important objects (together with their attributes and behaviour) for an application, CRC cards can be used. CRC cards were developed by Ward and Cunningham (1989) to document collaborative design decisions in an object-oriented environment. They have subsequently been used to help create the design since these cards can be used to model classes (Booch et al, 1998). The responsibilities or obligations of a class can be translated into a set of operations and attributes.

To return to the proof of concept phase, a scoping diagram was built, in consultation with users, showing the selected customer management sub-system at the centre and arrows, representing interfaces, to other

subsystems/actors. Then, from a prior user requirements analysis, six high level processes were identified and the links between them were shown in a context diagram.

The detail of each of the six processes was modelled using role activity diagrams (RADs), following Huckvale and Ould (1994). For each high level process defined, RADs were produced by walking through all stages of each process with the users. "As is" RADs (how things work now) and "to be" RADs (how things will be done in the future) were produced. Several member of the organisation were seconded to work full time on the project and so were available whenever required by the analysis team. Triggers, activities and states were shown and the business analyst observed that users found such diagrams easy to read. A goal was associated with each RAD which enabled a check to be made that the specified activities met that goal. There were various problems, however, that occurred when using this notation according to the business analyst. People tended to think about what they would like rather than the current systems and the "as is" diagrams and "to be" diagrams were virtually identical. The value of the "as is" diagrams was rather limited.

Problems arose because of the inexperience of the group. The primary and secondary scenarios were not identified. This meant that the users had to work at a high level of abstraction when modelling the detail of the processes. In these circumstances, the business analyst commented, *"It was difficult to get people thinking thoroughly enough."* A great deal of useful detail, it subsequently transpired, was omitted. Furthermore the roles of users were not sufficiently teased apart and one group of workers who appeared to carry out the same tasks had later to be separated into two. Nonetheless, producing the RADs helped the team understand the business.

The use cases were identified and the use-case diagram built after producing the RADs. The business analyst believed that it was essentially too late as, by this time, the process view was dominant and it was difficult to come in with a fresh perspective. The use cases should have been developed earlier, the analyst believed, when the high level processes were identified. The analysis team debated at length the problem of handling a process-based system within the O-O paradigm. It was decided that the use case approach allowed commonalties to be identified where a process oriented approach uncovered differences. Both were required. One possible solution, the use of activity diagrams to model processes, was not investigated since this notation was not at that time incorporated into the Rational Rose case tool Nor did the literature indicate how these could be fully integrated into an O-O methodology.

After completing the use cases, the candidate objects for the class diagram were identified by the analysis group so that interaction diagrams could be developed. This was done by a textual search of the RADs which produced about 20 candidate objects. After a brief walkthrough, invalid objects were eliminated and the remainder put into a class diagram. Class Responsibility Collaboration (CRC) cards were then employed to allow users to select attributes and behaviour for each object. CRC cards, the business analyst believed, should also have been used earlier to help select/validate object selection.

Interaction (sequence) diagrams were then drawn. It became apparent at this point, according to the business analyst, that *"we needed more information about which attributes were passed on from the objects at what point. So it was not enough for us to know that an object had 5 attributes."* Information was needed about which attributes to update at particular times. This information could have been obtained by using CRC cards again but time did not permit this. Interaction diagrams for this application proved to be very confusing as there were so many exceptions to be handled. The class model was, though, refined in the light of this exercise as were the use cases. Eventually there were only two principal use cases with various extensions. These formed the basis for the written specifications which were handed over to the designers.

With regard to interface design the approach referred to as lo-fi prototyping (Rettig, 1994) was employed. Instead of building a system to demonstrate the interface, lo-fi prototyping is paper based. A paper prototype of the interface is developed with a user playing the role of the computer. For the proof of concept project, users walked through a particular scenario following this approach. This was done, however, with a very limited goal which was to find out what knowledge had to be captured where. Someone was chosen to act the part of the computer and handed out the attributes as indicated by the scenario. This proved to be a very valuable exercise as users were able to provide even more relevant detail. This process was not carried out early enough, however, for this detail to be included within the analysis. The business analyst believed that walkthroughs of this kind should be carried out much earlier on. Finally, with regard to the interface, state transition diagrams were used to model volatile objects. There were not many of these, however, in this transaction processing system. The sequence in which the outputs were produced during the proof of concept phase and the dependencies between them are shown in Figure 2.

A major problem occurred, however, once the class diagram was handed over. The designers wished to minimise the use of inheritance. Consequently according to the business analyst *"We had a huge political debate at this time."* The decision was made that the designers would make a limited use only of inheritance and roll the objects up, that is amalgamate inheritance hierarchies such as customer, customer type etc, in their own version of the class diagram. This enabled them to have fewer objects, each with a large number of

attributes and methods. They were able in this way to reduce the number of messages passed. The detailed structure provided by the analysts, however, had been necessary to validate business rules so it had not been a waste of time drawing it up. Where a problem did arise was with the interaction diagrams which contained information useful for system development. The interaction diagrams, however, were linked to the analyst's class diagram and so the designers either had to lose this information or adapt the interaction diagrams to match their own version of the class diagram. Another problem which faced the designers was having to match up the attributes with those in the logical data model (produced at an earlier stage.) Only a very small part of the system specified was successfully implemented.

Overall, the business analyst felt that "*the most important thing I learned was that there is not any mature O-O methodology.*" She felt that during the proof of concept phase she was effectively doing research to find a methodology that could be used for a process-oriented application. She realised that there had been major problems with the sequencing of the stages in the methodology. For instance, use cases and CRC cards were not used early enough. Moreover, the full power of a technique such as CRC cards was not employed because of a failure to understand its potential. These problems occurred because all of the members of the team had a background in structured systems analysis and were probably too wedded to a process view and process-oriented techniques.

Use cases, however, she believed were vitally important. They assisted with identifying roles and actors, segmenting functionality and allowing for both components, such as screens, and code to be re-used. In the future she would suggest that they be built earlier in the proceedings.

THE ANALYSIS PROPER

The third interview took place after the first stage of the analysis proper. For this phase of the project UML (3.5) was the standard followed and the tools used were Rational Rose 98 and Requisite Pro version 3.0.7. Even at this point the team was "*still a bit confused about how they were going to marry business processes with use cases.*" However, advice from an expert in O-O allowed this critical issue to be resolved. The team followed the process he suggested with some additions of their own.

First, a context diagram was produced which showed all the actors outside the business and their interactions with it. From this, 6 high level processes were identified for the business. No links were shown between them. Instead, for each process, the associated business cases were specified. In one case there were 40 of these. In effect, a hierarchy of use cases was built although this, as the business analyst observed, is not a solution generally accepted in the literature. The top level (the 6 processes) had to be supported by the package facility of the Rational Rose case tool. The only problem that arose was that packages do not interact with actors. All the use cases were named and any relevant interactions between the actors shown in a use case diagram, again drawn in using the Rational Rose case tool. All this was achieved in one week.

The team was also advised to produce the detail of the business use cases first and, based on these, build system use cases. Teams were set up composed of users seconded to work on the project and analysts. These had the responsibility of developing the business use cases: writing the associated descriptions (scenarios) and identifying flows, alternate flows and exceptions. A clear distinction was made between the actions of the actor and those of the organisation. Only three weeks were allowed for this stage and the business use cases were not completely filled out. The intention was to use them to derive system use cases and not to become swamped with detail. The sequence of actions involving the actor and the organisation was shown in a two column table (Figure 3). One column showed the actor's action and the other the business's response including the title (Membership Services officer) of the person who carried it out. This idea worked so well that users were able to complete the relevant sections themselves and just bring them back for review. As the business analyst remarked " *These tables focused people's minds on who was doing the action and that you are going to get responses.*"

From these business use cases, it was possible to move directly to system use cases which followed the two column format previously described. In these, the workers within the organisation become the actors. Their activities are now shown in the left hand side of the table with the system activities on the right (Figure 4). In system use cases the actors act directly with the system. These system use cases included all relevant data but did not include interface details such as "Open Window." This was left to the interaction diagrams. To avoid incorporating many alternative courses of action per use case, each separate option was specified following the "Extends" feature of UML which allows multiple scenarios per use case.

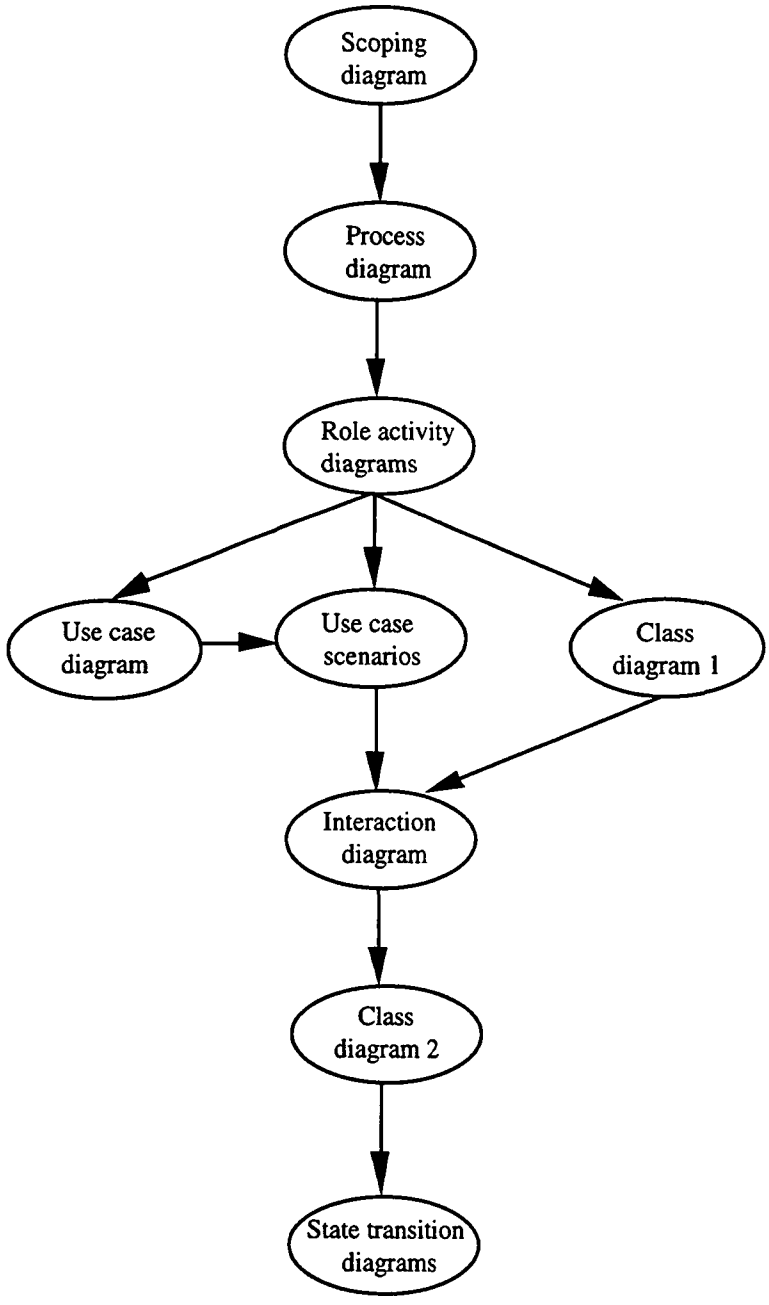

Figure 2: Outputs for the proof of concept phase

The business users (ten of whom were seconded to work on the project) assisted by members of the analysis team went through the business use cases, constructed the system use cases and added any necessary detail. This information came from various sources such as the current staff procedures manual and from the detail incorporated within existing software documentation. For the use cases which handled extensions to the existing system, members of the organisation brainstormed together with analysts on the team and built decision trees of the knowledge to be incorporated. Managers who had to sign off the use case specifications also added further information. The only problem was that some users were out of touch with what was happening within the organisation. As the analyst commented *"It's nice having users totally accessible to you, sitting by your side but because they have been out of action for at least a year it is not ideal."* The process of building the use cases would have been completed more quickly if the users had had more recent experience. Nonetheless the managers who signed off the specifications were always *au fait* with the situation and could make suggestions when necessary.

Once the system use case specifications had been signed off it was necessary to include relevant calculations within them. A team of programmers went through the code of the existing system and documented all of these. When incorporated within a systems use case, the calculations became a list of system actions or a "Uses" use case if the calculation was complex or used by more than one case. The business analyst remarked that *"Our use cases are becoming quite different beasts now from how they were before we put this detail in. It is not just this interaction between the actor and the system any more."* Several steps may be involved in the response to one worker action.

Actor	Organisation
1. Customer submits change of address	2. Membership Services officer updates address
	3. Membership Services officer sends confirmation
4. Customer receives confirmation of address change	

Figure 3: Flows in a Business Use Case

Batch processing also had to be handled within this framework. All the batch processes were represented by a use case with a list of system actions associated with an actor referred to as "Father Time. "
All in all, there were more than a hundred use cases. Some of these were for essentially non-functional requirements such as the setting up, for security purposes, of access rights. Auditing requirements such as those for performance or disaster recovery were described in a separate Requisite Pro table as a "Non-Functional" Requirement Type.

System Actor	System Response
1. Membership Services Officer enters member number	2. The system displays the customer details including address
3. Membership Services Officer enters new address	4. The system requests confirmation of the change
5. The Membership Services Officer confirms the change	6. The system updates the customer record and prints off a letter to the customer confirming the change of address

Figure 4: System Use Case

The signing-off of the specifications was not the final stage of the analysis. It was still necessary, in another iteration, to complete the specification of the "Uses" use cases, that is to identify common functionality. The team had been advised to leave this to a comparatively late stage so that they fully understood the business and avoided coming to premature conclusions. Finally, state transitions diagrams were used to show particular sequences of actions that would have to be carried out by a staff member. This notation was only employed since there was no support in the Rational Rose case tool for activity diagrams.
As well as taking part in the development of the context diagram, business and system use cases, the business analyst also had the responsibility of writing the guidelines for developing the use cases.
The business analyst was not involved with the building of the class diagram but did participate in two reviews of it. The initial plan had been to derive the business object model from interaction diagrams drawn for the core

use cases, the top 20%. These core use cases were those most critical to the business and incorporated the major functional requirements. This process proved very tedious and was abandoned. Instead a CRC card approach was used to build the business model. A team walked through the 30 selected use cases and produced a CRC card for each object. In this way reasonable cover was provided. This diagram, built in the Rational Rose case tool, showed the relationship between objects, attributes and methods. It was only expected at this point that 80% of the objects would be identified. Other objects and methods would be determined later when the other use cases were considered. The class diagram would, in the future, have to be matched against the logical data model.

Once the object model was in place, interaction diagrams could be built to link actions to classes and also show interface details such as "window." Coding could then commence with all the necessary detail present in the system use cases. The outputs for the process described above and the order in which they were produced are shown in Figure 5.

Overall the business analyst now felt confident that there was a way to handle a large scale transaction processing system in O-O. "*Analysis paralysis*" had been avoided by the iterative nature of the process used for O-OA. She re-iterated, too, her opinion that system use cases were invaluable for system specification, allowing the relevant parts to be sliced off for implementation by programmers. Nonetheless, there were strengths, she believed, in other approaches. The business analyst thought that it was due to her data modelling background that all relevant data attributes were incorporated into the system use cases. This was important in a transaction processing systems.

DISCUSSION

In this section, the experiences of the business analyst and the organisations described above are reviewed in light of the literature on migrating to O-O technology. To summarise, the business analyst emerged from the fifteen months spent working on O-O projects feeling quite confident of her ability in this area; the various pieces of the jigsaw coming together to form a whole. During this period she had read widely in the area and attended courses on both UML and the Rational Rose case tool. She had also had the benefit of discussions with a consultant in the area of O-O. Lengthy debates with analysts and designers also helped to clarify many important issues. Ambler (1998) listed many ways in which an individual could improve their knowledge of O-O technology: reading books, magazines and journals on the topic, attending courses and conferences, browsing the internet and obtaining a mentor. The analyst (whilst not aware of the Ambler text) had carried out all of these activities save for obtaining a mentor. Unfortunately there was no-one available on any of the projects with the expertise to perform this role. This situation is quite common. Cockburn (1998, p 72) noted that "*Market demand currently exceeds the supply of properly trained mentors.*" Whilst not ideal, the situation was alleviated by extensive interaction with other analysts, designers and the consultant.

Whilst Booch (1996) and Cockburn(1998) both comment on the time taken for programmers to become effective using O-O techniques, it can be argued that analysts are in much the same position. Considerable time and effort are required. Business analysts have to have wide ranging expertise and need to be familiar with techniques from both structured systems analysis and the O-O paradigm for systems requirements. A synthesis of knowledge has to be achieved so that analyst knows which techniques to use for which projects or parts of projects.

Organisations, as Booch (1996) and Cockburn(1998) recognise, are faced with many problems when making the transition to O-O technology. These were particularly severe for the organisation that wanted to develop the large transaction processing system. More than two existing practices were changed (method, staff, tools, development language). According to Booch (1996) the outcome is uncertain in these circumstances. Nonetheless, progress was made on the project.

Many of the actions taken by the management on the large transaction processing system correspond to the suggestions made by Booch (1996) and Cockburn (1998). A pilot project (the proof of concept phase) was undertaken. An appropriate methodology was developed after outside expertise was brought in. An object-oriented model was built during the analysis phase. A process of incremental development was followed. This allowed a divide and conquer approach to be employed. First the use cases were identified and the business use cases were built. This was only done, however, at a level of detail that was sufficient to form the basis for building the system use cases. It was at this point that detailed analysis commenced. When the flows for the system use cases were finalised the required calculations were added. Non-functional requirements were also associated with them. Finally, the "Uses" use cases were developed. What was a very large task was made manageable by breaking the analysis process into several stages and progressively adding more detail. Iteration around the requirements allowed time to be set aside to re-visit parts of the analysis. The system use cases were checked by business managers and mistakes and omissions had to be rectified. The class diagram, too, was being developed iteratively, in at least two passes. Real users were involved from the start, although not all of them

were, by the time of the analysis proper, up to date with what was happening in the organisation. Finally, staff were provided with the appropriate training and tools.

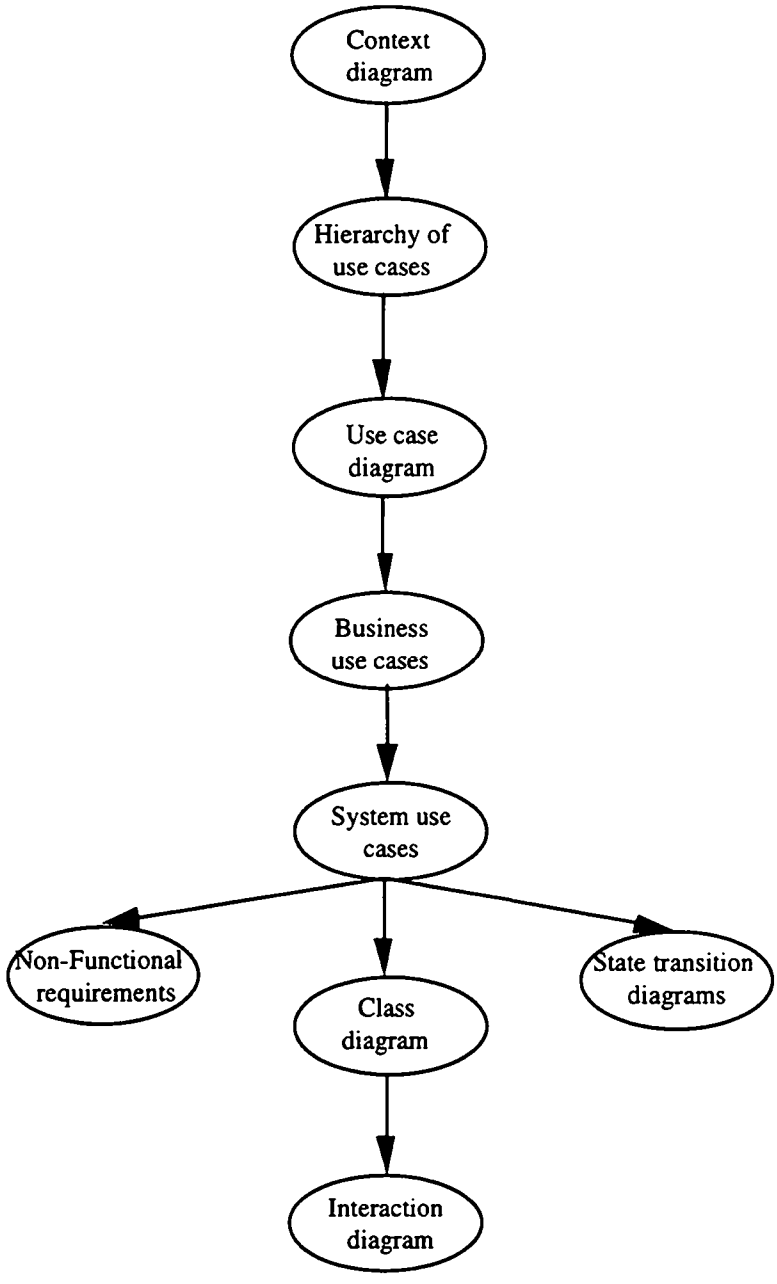

Figure 5: Outputs for the analysis phase

It is worth examining one of these issues in more detail, the development of an appropriate O-O methodology. The major difficulty that arises for a new problem is the selection of a suitable process. Tailoring a methodology from the many notations available in order to specify the solution is a non-trivial task. For the transaction processing application described above there were various features of the solution that do not arise in O-O texts (which often deal with real-time systems.) The differences include the hierarchy of use cases and the introduction of an actor such as "Father Time'. The system use case descriptions (scenarios) were somewhat unusual too, often involving a long list of system actions and a complex calculation in response to just one actor action. Developing the methodology for the project, however, took a great deal of time.

Initially, it was hoped during the proof of concept phase to finalise the process that would be used on the project. It became clear to all the members of the analysis team, during this stage, that the approach suggested was

unsuitable for O-O analysis. Team members had become *au fait* with the nature of business problem and various notations but had not brought together the different aspects of the O-O analysis process. At this point additional expertise was brought in. It is debatable whether bringing in the consultant earlier would have had much benefit. When attending the UML course, assistance had been sought for the proof of concept phase but only the use of activity diagrams was suggested as a possible solution. By the time of the consultant's arrival, the team had carried out a preliminary analysis and knew exactly what their problem was. The solution proposed, the hierarchy of use cases, is not one found generally in the literature and had to be tailored to their problem. With the major problem solved, the team were able, themselves, to deal with issues such as how to handle batch processing. It was vital, though, to obtain assistance in a timely fashion - too soon and the problem would not have been sufficiently understood, too late and the project would have failed at the analysis stage.

CONCLUSION

For the business analyst involved in moving across to the O-O paradigm from a data modelling perspective, the journey involved many stages. She learned first that a gradual move from structured analysis (data flow diagramming and data modelling) to O-O was fraught with problems especially with regard to the identification of objects and methods. A purer O-O approach was needed. Use cases provided this. She also realised that they should be produced earlier rather than later in the analysis as happened in the proof of concept phase. Of most importance, the business analyst discovered that the O-O approach selected has to be tailored to fit the problem. Descriptions of O-O usage in textbooks do not always map to real world problems. It was easy at the start of a large project to feel overwhelmed by the complexity. In the transaction processing system described above, there were functional and non-functional requirements to be considered. Batch processing was still necessary and had to be handled within an O-O framework. Seeing how the different aspects of the system could be specified in a way that allowed the whole to be comprehended was an invaluable learning experience. Overall, the analyst now feels that she can see the advantages and disadvantages of the various approaches with which she is familiar. She does not believe that her previous experience of structured analysis can be discounted even within the O-O framework. At times a process or data focus can prove very useful.

Just as individuals are having to come to terms with O-O, so are organisations. Changing over to the O-O paradigm is not a straightforward matter as many authors have pointed out. A large number of methodologies can be found in the literature and many case tools are available. Even the advent of the standard, UML, has not resolved this situation; there are still competitors around. Small projects are one way for an organisation to advance (see the prototype system). Another way of learning about O-O is to have a proof of concept phase. Some might argue that an organisation should not commence a large project without having in place people at every level with requisite experience. Unfortunately, due to O-O's lack of maturity, it is impossible to know in advance that the right people are in place. Success on projects of one kind (real-time) does not ensure success on others (transaction processing). Organisations have to appreciate that solutions may have to be tailored to the problem in hand. What is essential, however, is that when assistance is required it is made available. A divide and conquer approach is also recommended for larger projects. This fits in well with the iterative nature of O-O which allows the detail to be added in at later stages when the problem is properly understood. Staff training is another important issue for organisations migrating to O-O as it not only provides practice in the use of tools and notations but allows for discussions with those experienced with the technology. The business analyst was fortunate that she not only received adequate training but had the opportunity to develop her skills incrementally, moving from smaller to larger projects. This is an ideal situation which should as far as possible be replicated by organisations. An organisation should also endeavour to ensure that those starting off in O-O analysis have the opportunity to consult with those more experienced than themselves even if this expertise cannot be found within the organisation. It is a major challenge for organisations to recognise that their analysts may not be fully productive from the start but require time to adapt to the O-O paradigm. Given the current shortage of trained staff, a related difficulty may be that of retaining staff once they are capable of acting as mentors and consultants to others.

REFERENCES

Ambler, S.W. (1998) **Building Object Applications That Work**, Cambridge: Cambridge University Press.

Beck K. & Cunningham W. (1989) "A Laboratory For Teaching Object-Oriented Thinking", **OOPSLA'89 Conference Proceedings**, October 1989, New Orleans, pp 1-6.

Benbasat, I., Goldstein D. K. & Mead, M. (1987) "The case research strategy in studies of information systems", **MIS Quarterly**, 11(3), pp 369-386.

Booch, G. (1996) **Object Solutions**, Menlo Park, Ca: Addison-Wesley.

Booch, G., Rumbaugh, J. & Jacobsen, I. (1998) **The Unified Modeling Language User Guide**, Menlo Park, Ca: Addison-Wesley.

Cockburn, A. (1998) **Surviving Object-oriented Projects**, Reading, Mass: Addison-Wesley.

Goldberg, A. & Rubin K. S. (1995) **Succeeding with Objects**, Reading, Mass: Addison-Wesley.

Graham, I. (1994) **Migrating to Object Technology**, Wokingham: Addison-Wesley.

Hammersley, M. & Atkinson P. (1983) **Ethnography: Principles and Practice**, London: Routledge.

Hares, J. S. & Smart J.D. (1994) **Object orientation : technology, techniques, management, and migration**, Chichester: John Wiley & Sons.

Huckvale, T. & Ould M. (1994) "Process Modelling: Why, What and How" in **Software Assistance for Business Re-engineering**, Eds, K. Spurr, P. Layzell, L. Jennison and N. Richards, Chichester: John Wiley & Sons.

Keat, R. (1981) **The Politics of Social Theory: Habermas, Freud and the Critique of Positivism**, Oxford: Blackwell.

Meyer, B. (1995) **Object Success**, London: Prentice Hall.

Rettig, M. (1994) "Prototyping for Tiny Fingers", **Communications of the ACM**, 37(4), pp 21-27.

Walsham, G. (1995) **Interpreting Information Systems in Organisations**, Chichester: John Wiley & Sons.

White, L., Gamble, D. & Blunden, S. (1992) An examination of the epistemological foundations of an action-research project, Discussion Document, Faculty of Agriculture and Rural development and the Social Ecology Centre, University of Western Sydney, Hawkesbury.

Yourdon, E. (1989) **Modern Structured Analysis**, Englewood Cliffs, N J: Prentice-Hall.

Yourdon, E. & Argila C. (1996) **Case Studies in Object-Oriented Analysis and Design**, Englewood Cliffs, N J.: Prentice Hall.