ELECTRONIC **WORKSHOPS IN COMPUTING**

Series edited by Professor C.J. van Rijsbergen

Gerard O'Regan and Sharon Flynn (Eds)

1st Irish Workshop on Formal Methods

Proceedings of the 1st Irish Workshop on Formal Methods, Dublin 3-4 July 1997

Paper:

Presenting Object Oriented Formal Requirements Specifications: Insights From An Action Research Study

Danielle Fowler and Paul Swatman

Published in collaboration with the
British Computer Society

BCS

Springer

Presenting Object Oriented Formal Requirements Specifications: Insights From An Action Research Study

Danielle C. Fowler
Monash University
Melbourne Australia

Paul A. Swatman
Swinburne University of Technology
Melbourne, Australia

Abstract

This paper describes, in outline, our experiences in structuring and presenting formal requirements specifications, and in particular the insights gained from an action research study undertaken within the Western Australian state government. The overall aim of the project was to evaluate and enhance an information systems development method, known as FOOM (Formal Object Oriented Method). FOOM was synthesised from research into:

- the object oriented approach
- mathematically formal specific ation languages
- socio-organisational contextual analysis.

The paper focuses on a study of the communication between the specifier and the client—principally on the problems associated with specification validation—and the effect this had on the structure of a FOOM requirements specification document. The result of the study described was a specification structure which allows precise communication both between developers/designers and the less mathematically sophisticated users of the system. While the focus of the research was validation of specifications written in the formal specification language Object-Z [10], we argue that the results of our analysis are of importance in the elicitation, refinement and validation of requirements specifications whenever a formal modelling approach is to be adopted.

1 Introduction

Poor quality systems and software development permeate the information systems industry. System development projects are notorious for being unpredictable and expensive, and for failing to meet requirements when they are delivered. A significant contributory factor in this is the "requirements" problem. Within the software development community there is general agreement with Boehm's [1] view that most software errors are introduced during the specification stage and that it is much cheaper to fix problems during the early stages of software development than at the end (see e.g., [9,22,35,36]). Unfortunately, arriving at a complete and consistent specification is perhaps the hardest part of the software development task [3]. Nonetheless, identifying and stating the requirements accurately and precisely is critical to producing good quality software. It is widely held that specification, or more generally requirements engineering, will continue to be a critical issue throughout the 1990s [15,26].

Formal specification languages offer the potential for precise, unambiguous statements of requirements. Originally valued for their facilitation of formal verification, formal methods have more recently come to be acknowledged for their contribution to problem understanding [16,27,30,37]. We were interested in investigating their potential for enhanced communication within the requirements elicitation and specification process, and for their ability to aid evaluation of understanding.

The potential benefits of applying formal methods within the information systems arena, in the

system development methodology (ISDM) called FOOM (Formal Object Oriented Method). FOOM is an approach to high-quality requirements engineering which incorporates elements of the soft systems [4-6] approach with formal specification and the object oriented paradigm. We believed strongly in the potential benefits of a method synthesized from these areas. We recognised, however, that FOOM needed to be not only theoretically sound, but useful and applicable within its intended target domain — the commercial IS aren a.

For this reason much of FOOM's development took place within an action research project conducted within the Western Australian State government. In this paper, we discuss the conduct of the study, focussing on:

- difficulties associated with communication between mathematically unsophisticated clients and the development team
- how models of the problem context (loosely, requirements specifications) developed using the evolving FOOM the method came to be validated
- describe how the research process led to the identification of the need, in the context of specification validation, for a mechanism to allow us to illustrate the behaviour of a system modelled within the object oriented paradigm
- discuss the development of such a mechanism—a diagrammatic notation designed to aid in validating system behaviour, though we do not describe the resulting notation in detail [1]
- draw conclusions about the likely generality of our results.

2 Evaluating FOOM in a Commercial Setting

The FOOM framework was developed by synthesis and logical argument, drawing on research in a number of largely independent areas across the breadth of the information systems and software engineering domains [30,31]. Preliminary evaluation of feasibility and potential benefit were undertaken by means of simulated [32] and small commercial [33] system specifications; and by means of educational case studies and pseudo-laboratory experiments [29]. FOOM was then extended further through an action research [28] study undertaken at the Western Australian Government Department of State Services. In the course of this study, we applied, evaluated and enhanced the framework. A process model for the resulting method is depicted in Figure 1.

We adopted an action research approach to this phase of our project with the intention of strengthening the validation offered by our earlier work, while retaining the opportunity for the expected necessary enhancement of the development method. We felt that the most significant risk was of impracticality, not of theoretic weakness. The aim of this study was to gain some further confidence that:

- formal specification could be adopted within the information systems domain — that is that it was possible to overcome the well-documented (see, for example, [8]) barriers to adoption of a mathematical approach:
  - a perceived lack of value due to beliefs that formal methods would not scale up to real world applications and/or that they were only relevant to safety -critical systems
  - inadequate mathematical capabilities of the professional community
  - immaturity of formal methods
  - a fear of intellectual inadequacy
- an approach based on FOOM could be applied beneficially to conventional, reasonably-scaled problems within the information systems domain.

In designing our study, we sought:

- a reasonably conventional information systems department. Most practical applications of formal methods have been in association with unrepresentative organisations such as Praxis and Logica Cambridge, which are widely acknowledged as elite software engineering organisations; and IBM's Hursley Laboratory

[1] The interested reader is referred to [14] and [34].

- a large-scale system development to demonstrate that the approach would scale up to real-world problems

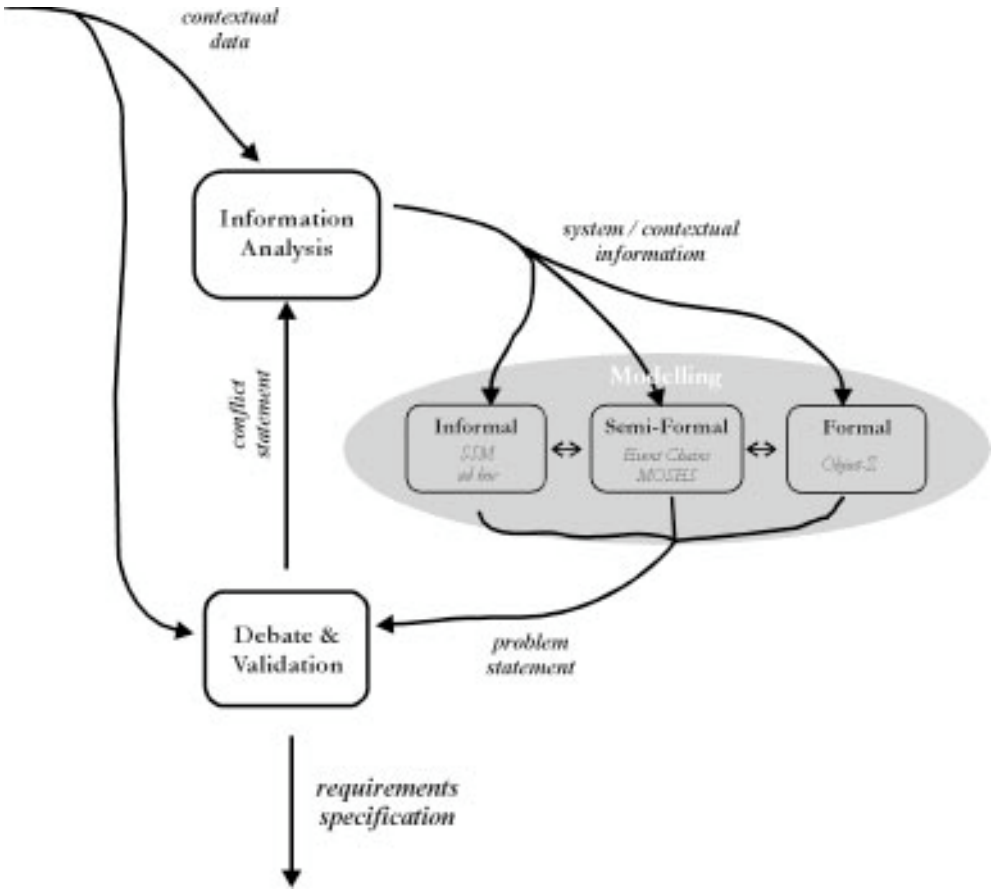

Figure 1: The FOOM Development Process

- an unequivocally information systems development with non-technical clients to demonstrate that benefits extended beyond the technical domain
- a development which was important to the organisation (to ensure that the clients were interested in achieving a high-quality result) but not time-critical (since we recognised that, since we had to conduct analysis of the process in addition to the development itself, our approach was likely to be somewhat labour intensive
- an organisation where we could find a high-level "champion" for our study.

We were fortunate to gain agreement from the Western Australian Government Department of State Services (DoSS) to allow us to undertake the study in relation to the SupplyNet project — a redevelopment of a whole of government Electronic Data Interchange (EDI) system undertaken in collaboration with Telecom Plus and Ferntree Computer Corporation. This was a particularly good basis for our study since:

- the information systems staffing was representative of large organisations, in the case of DoSS and Telecom Plus and of IS domain software houses, in the case of Ferntree
- the project fitted all our requirements — and more. SupplyNet was a multi- organisational systems development with participants in three States. Thus, precise communication was highlighted
- we had the backing of the Director of Information Technology at DoSS.

In the next section we describe the iterative development of the FOOM approach to the validation of specifications, then describe the effects this had on the structure of our formal specifications for presentations to clients.

3 Understanding and Modelling Problem Domains

In common information systems development practice, a problem context is first informally and then systemically specified. The informal specification is ambiguous, imprecise and contradictory. Various systematic specification techniques[2] have been designed to reduce this ambiguity, imprecision and contradiction. Unfortunately, while systematic techniques assist in this process, only specifications written in formal languages which have a well-defined syntax and semantics are capable of precise analysis.

Once the solution design and implementation have occurred, the formalised information system which results has a well-defined behaviour, free of ambiguity. Programming languages are, in a sense, mathematical specification languages and a program listing (in, for example, C or COBOL) is, therefore, a precise and complete specification of the behaviour of the corresponding executable — all programs do something, the question is whether or not this something happens to match the intention of the specifier.

In essence then, the systems development process takes an ill-structured and possibly contradictory context as input and delivers, inter alia a precise, well-defined (but, too often, neither the desired or expected) computer system as output. The critical issue, therefore, is to determine at what point we should resolve the imprecisions, ambiguities and contradictions of the problem context. The FOOM approach, which is illustrated in Figure 1, is to resolve these issues during the information analysis process. The argument, set out in detail in [13] and [30], is based upon the following assertions:

- we must formalise our approach sometime (a program is mathematically precise — if somewhat difficult for human beings to analyse)
- it is significantly more likely that the participants in the information analysis process (which would include actors within the context) would bring problem-context-related worldviews to the identification and resolution of ambiguities and contradictions than would software designers and computer programmers whose worldview is likely to be essentially technical
- a formal specification of requirements may be communicated to the designers and programmers precisely and unambiguously.

4 An Evolving Approach to Validation

During the two and three-quarter years of the action research study, a number of formal specifications were developed of different aspects or components of SupplyNet. The exploration and resolution of the problems associated with validation formed a significant part of this process. Each successive specification provided clearer indications of the most critical aspects of the problem, enabling us to fine tune our approach. This evolutionary process is described below.

The specifications developed were of quite different aspects of SupplyNet, ranging from a reasonably technical access control specification to the purely organisational process associated with registration and training of new subscribers to SupplyNet. The information gathered in the process of their development therefore came from a wide variety of people from within DoSS, Telecom Plus and Ferntree.

4.1 Introducing Formal Specifications to Clients

The first specification developed was a reasonably small, high level specification of SupplyNet access control features. It was developed first for two reasons:

- as a simple specification of processes which could easily be understood by DoSS, it was likely to provide a `gentle' introduction to Object-Z (the formal specification language used) in terms of presenting mathematics to the people involved with validation. Only one person from DoSS — the SupplyNet coordinator — had attended the first two lectures of ten-week introductory course on Object-Z, which covered the basics of set theory and predicates.

[2] For example, Dataflow Diagrams [10].

- it was envisioned that it could be developed fairly quickly in order to provide DoSS with early reassurance of the benefits of the project (and provide us with indications of the feasibility of the research).

We selected one of the various alternative approaches to access control through interviews with the SupplyNet co-ordinator, whose focus was on the business rules/user requirements, and various members of the IT department within DoSS, whose foci (as system developers and implementors) were more technical in nature.

Once a draft specification had been developed, we conducted a walkthrough in order to validate it. The people present at the walkthrough included the SupplyNet co-ordinator Arthur Wilson, his assistant/junior, and the person in charge of supporting SupplyNet within the IT section at DoSS.

Our approach to the presentation of formal specifications at this point in our research was that traditionally used within the scientific/technical domain, with explanatory text used to supplement the mathematics. It was found that the construction of ad hoc informal diagrams during the explanation process aided in understanding. The diagrams worked best when made as SupplyNet-specific as possible (in terms of terminology and detailed functionality) even if this made them more complicated than needed, because it put the explanation into a familiar frame of reference.

The SupplyNet co-ordinator was receptive to the mathematics involved, although he acknowledged that his acceptance of the specification was largely based on the explanation associated with the presentation — he didn't have sufficient confidence in his understanding of the mathematics at that point, and the supplemental text was of little use because it described “the maths” rather than “the system” as he understood it. As it was at a very early stage of the research and our user acceptor was still becoming familiar (and comfortable) with formal specifications, it wasn't possible to draw any firm conclusions as to the effectiveness of this approach, although it seemed that some kind of additional diagrammatic support would be helpful.

The other two participants also found the system design useful and comprehensible. They had not been trying to understand the mathematics fully as they had not attended the introductory course. They understood the system in question — SupplyNet — very well, however, and had been attentively comparing the “system” as described in the specification with the existing one, and also making sure the explanation was consistent within itself — which required an understanding of the model presented. An important outcome was that all participants accepted as valuable the presentation of the mathematics, an issue about which both we and they had initially been concerned.

Although inconclusive in themselves, the results offered encouraging preliminary confirmation of our theories. They demonstrated that:

- a formal specification could be presented to IS personnel without mathematical backgrounds; the fear of intellectual inadequacy could be overcome
- a specifier with a conventional IS background (a non-traditional background for formal specification development) could develop and present a formal specification successfully; in fact modelling skills (and the ability to explain models) are more important than a strong mathematical background.

  potentially, at least, the mathematical specification could be incorporated within the requirements validation process.

4.2 Round 2: An EDI specification

The second specification presented to DoSS was an existing formal specification of a generic EDI communications system (Swatman and Swatman, 1992a). This was undertaken because a problem was identified within DoSS which we believed could be illuminated by the specification.

One of the functions available in SupplyNet is the facility to send purchase orders. At the time, these purchase orders were sent as structured free text. As part of the planned progression of SupplyNet towards the adoption of EDI, SupplyNet was being modified to enable a specific group of government users, namely those departments using the existing Common Use Purchasing System (CUPS), to send purchase orders in standard ANSI X12 format to those of their suppliers which were EDI-capable. After consultation with Telecom Plus, DoSS decided to use a cutdown version of the ANSI X12 purchase order (document 850) tailored specifically for the needs of the CUPS users.

Upon discussion with the SupplyNet co-ordinator it became evident that due to his business (rather than IT) background, some of the ramifications of this decision were not immediately apparent to him. This presented us with an excellent opportunity to test our approach further:

- the problem was sufficiently small that the specification was manageable without being trivial
- the co-ordinator of SupplyNet was essentially a business analyst — he had an understanding of the business environment in which SupplyNet operates, but had limited IT experience. This made it possible to gather information on whether a more complex formal specification could be made accessible to someone working with an information system with minimal exposure to either mathematical theory or the techniques themselves; it was also our intention to observe any changes in his attitudes and impressions over a period of time
- more information could be gathered about whether someone with no mathematical background and minimal experience with formal specification techniques could achieve a level of proficiency sufficient to enable them to help someone with no experience with the techniques to understand a more complex specification (and to what degree)
- the specification would enable us to compare the outcome with those of the first specification, which would provide us with added confidence in the results.

The decision, as seen by DoSS, was between using the full ANSI X12 purchase order and a (cheaper) especially cut down subset. In fact there are three alternative situations reflected in this choice:

- the complete ANSI standard document could be implemented, with any unnecessary detail simply hidden from the user by the program interface. In this manner the document may be sent straight to any supplier using the X12 standard
- the use of a cut down version of the document gives rise to two distinct alternatives:
    - each supplier dealing with the CUPS users could adapt their own system(s) to handle the atypical document; or
    - the network vendor could convert the CUPS version of the X12 purchase order to the ANSI standard.

The first alternative in this case was simply not acceptable while the second meant a conversion cost which would likely have been borne by the individual departments using SupplyNet.

Our objective in presenting the specification was more clearly to illustrate the differences between the three alternatives. After explaining our intentions in presenting the specification and assuring the SupplyNet co-ordinator (the validator) that it was not necessary to understand the entire specification to grasp the points we wanted to make, we explained the system using the same approach as for the first specification (i.e. describing the different mathematical components in the specification — the sets and types, the “objects”, and the operations within the system).

The coordinator asked questions throughout, wherever he was unsure of the basic intent of the various operations. The use of simple diagrams to explain the various sets, functions and relations again proved to be very useful. After finishing presenting the parts of the specification necessary to illustrate the problem (which was the majority of it, in varying degrees of depth), we asked the coordinator if he could see why the specification showed it to be impossible for a cut down CUPS purchase order to be received and treated by a supplier as an ANSI X12 purchase order without conversion.

He found it helpful to draw diagrams on a whiteboard in order to confirm that he understood things properly. While the specification explained and “proved” the situation to the coordinator, it could not be easily used by him to convey the argument to other people, hence the need for him to translate the concepts into a form that could be readily explained by him and easily digested by others.

The most obvious problem that the coordinator encountered was that of regarding the system in terms of sequential procedures, rather than in terms of objects and states. We found that:

- the whole process was useful to the coordinator because it helped him to understand how an EDI communication system works (at a level of detail to which he was not usually exposed). It clarified the “problem area” in addition to the specific questions of interest
- the specification, together with some of the diagrams, made things much clearer for him than the verbal explanation we had originally provided
- he suggested that as it was his first experience with a complete specification — and of a system with which he was not terribly familiar — he needed to see how the entire system fitted together. He

believed, however, that in future a focus on relevant sections of a specification would suffice, if the specification described a familiar context. This was an important result because it showed his growing acceptance/belief in the value of a formal, mathematically based specification (that is, his greater confidence in such specifications than in other IS specification techniques previously encountered) and his increasing competence in their use.

5 The Switch to Object Orientation

At this stage it was decided that the object oriented extension of the Z language, Object-Z, would be a more appropriate choice of specification language. Once specifications became more than trivial in size, some sort of structuring mechanism was clearly needed, to make both their development and their presentation more manageable. The type of informal diagrams which we had found to be of use with the earlier specifications suggested that the OO approach might be beneficial, and the relatively small difference in notation between the languages meant that our participating users would need to learn only a few additional symbols and concepts rather than a totally new specification language. This change over would also allow us to test claims that the object oriented approach offers a more natural paradigm for understanding.

We decided to redevelop the first specification written for DoSS in Object-Z, as well as develop a third specification of the process of registering new suppliers subscribing to SupplyNet. DoSS requested that this specification be undertaken because the then informal registration procedure was resulting in errors and omissions. The specification was interesting for quite a number of reasons, including the fact that it was a purely organisational information system — no computerisation was involved. Most significantly, it was a considerably more complicated specification than any of those previously developed.

There were also significant changes made to the presentation of the requirements specification document. It was becoming clear that different audiences existed for the specification document, and their requirements, as well as their level of knowledge of object orientation, formal methods, and the application domain, would be quite different. The different audiences for the specification were divided into a number of classes:

non-computing users These are typically the management staff responsible for initiating the requirements investigation or those who have contributed to its development. They need to understand the system, but not necessarily the mathematics involved

specialist users These are the users responsible for the acceptance of the requirements specification itself and hence must understand the specification fully in order to validate it. Their interest is in comparing the problem situation they understand with the representation modelled

designer/implementors These are the development staff responsible for taking the requirements document and producing (if required) a computerised information system to match. As they may have no understanding of or exposure to the problem situation, their focus is on understanding the particular model of requirements given to them and the technical issues associated with implementing it.

Given this breakdown of functions, several possible alternatives suggested themselves for the breakdown of the documentation:

- the first was to couple a high level description of the object oriented structure of the specification with the corresponding Object-Z class definitions. The advantage of this approach was that it would be easier to structure the specification into discrete sections. The disadvantage was that different levels of explanation would be needed depending on the OO experience of the audience. More tutorial style information would be required for those without OO exposure, as well as a separate section for presenting a more technical OO explanation (such as association and aggregation diagrams). Too much tutorial information would be redundant for those familiar with OO, and too much technical information would be confusing to those unfamiliar with OO. It seemed a blend would possibly work: assuming minimal knowledge of OO (the most likely case), a brief description of relevant object and class structures and what they represented within the problem domain could be presented first, then the detail of the Z could be combined with a middle-ground explanation which concentrates on the functionality of the system. The rest of the OO detail could be included as an Appendix
- the second was to separate completely the OO description from the Object-Z, with a traditional OO description comprising static OO diagrams to explain the structure coming before the formal

representations. This would alleviate the problem of breaking up the Object-Z class descriptions, but involve a large duplication of information as much of the structuring information is common to both representations

- the third alternative was top-down versus bottom-up. Both management and specialist users seemed to prefer a top-down explanation, while designers seemed to prefer bottom-up. A choice between one or the other, or both, needed to be made.

The two specifications were presented (separately) using the same approach. Rather than allowing the structure of the presentation to be dictated by the order of the mathematics within the specification, we presented each specification in terms of the objects and classes within the system, and explained how they communicated with one another. Object oriented diagrams were also used to illustrate the structure of the specification. This approach was considered by the users to be more successful, primarily because it presented the functionality of the system in a more "natural" order. It was more beneficial to indicate the overall structure of the system from the beginning (a top down view), than to build up a picture slowly by examining the lower level structures within the system (a bottom up approach). It was easier also to break larger specifications up into manageable portions for explanation Although it was a more successful approach to validation than our initial attempts, it became clear, particularly with the larger registration specification, that this approach still had a number of deficiencies. Contrary to the prevailing wisdom, users responsible for validating specifications do not, in our experience, think naturally in terms of a set of objects comprising a system — they think in terms of the behaviour which makes up the system, which may be provided by a number of operations spanning multiple classes. Describing objects together with their associated behaviour is a useful approach from the point of view of developers who are responsible for implementing a system, although it seemed to be of more limited value in the information and requirements analysis phase.

It seemed clear from these results that some better way of presenting and explaining the requirements specification was required. Clearly, the greater the understanding of the specification by the user acceptors, the greater the confidence which can placed in the validation. It was critical therefore to find a way of presenting specifications which facilitated the best possible understanding of the specification.

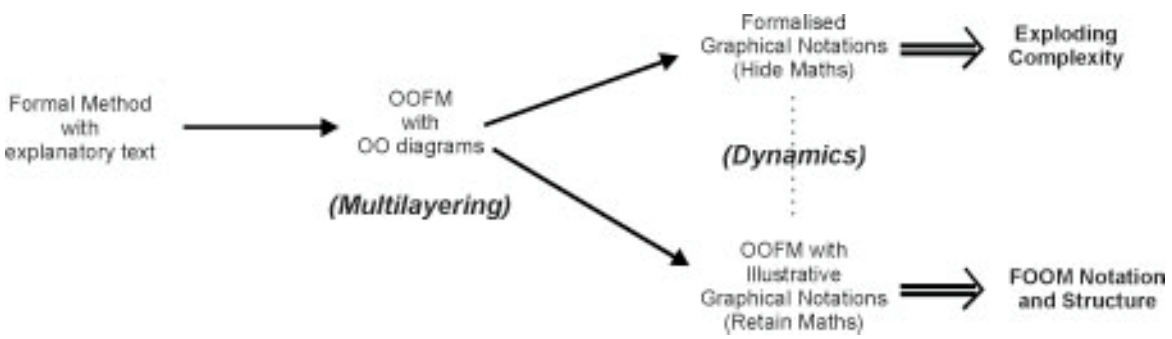

Figure 2: The evolution of the FOOM notation and structure

Figure 2 depicts the evolution of our approach to validation and the contribution of each specification developed to this process. Given the results we had obtained from the specifications developed so far, it seemed clear that simply supplementing a formal requirements specification with explanatory text was not sufficient; nor was adding static OO diagrams to this combination. What was becoming clear was the need to layer a specification into different levels of abstraction. In this way the complexity contained in the specification could be managed. OO supported such a structure well, although a way was needed of presenting the OO specification with different levels of detail. Rather than having all of the detail in the specification visible at the one level (on a class by class basis), we needed to be able to show detail in the specification only when appropriate to the explanation.

It was evident that the problems of validation we were experiencing went beyond the characteristics of FOOM or formal methods — the issue was one associated with OO in general, that is, is OO a natural way of modelling a system, as is often claimed by the OO community?

This problem can be clearly seen in the example of a car. Following a traditional OO approach, we might describe:

- the car as a collection of sub-objects — such as an engine, a set of wheels and so on
- the behaviour of the car (by means of a state transition approach, or a derivative technique, we may describe the way a car will change state, accept inputs and produce outputs in response to the

occurrence of events. It is most common to organise the presentation of information about such dynamic behaviour around each class — that is, all possible dynamic behaviour of a car will be collected and presented together — although Jacobson et al [19], for example, have suggested structuring the presentation of dynamic behaviour around the event which gives rise to the behaviour.). For example, we may say that the car responds to a “go faster” message — the input being pressure on the accelerator, the output being the new reading on the speedometer and the associated state change being a revision in the speed of the car object. It would also be legitimate to describe dynamic behaviour at a more primitive level, focusing, perhaps, on the behaviour of the accelerator pedal or the fuel injection system.

This presentation is clearly suitable for those who will design and build the car. Encapsulation and information hiding have been applied in a manner which corresponds with the cognitive models which the OO world has argued are appropriate for implementors. The presentation is also suitable for a prospective driver of the car who is protected from any need to”look under the bonnet”. There is, however, an important interest which is not well served by the cognitive model which underlies the approach to encapsulation and information hiding adopted within the presentation discussed. The interest is that of the person responsible for licensing the car for use on our roads. The licensor is interested in the interplay of components within the car as it implements some ``user level'' behaviour. From this perspective, we want to show that on pressing the accelerator, fuel will be pumped into the carburettor and then passed into the engine, resulting (assuming the car is in gear) in an increase of speed.

In other words, the licensor perceives the car to be sets of related events which combine to describe the (system level) behaviour of the car. The analogue of the car licensor, in the world of information systems development, is the user acceptor — the person with whom the specification is validated.

5.1 The role of mathematics in the validation process

We now decided to question one of our underlying assumptions — that is, whether actually showing the mathematics involved was indeed necessary. We had had success with the use of informal diagrams, which seemed to facilitate understanding better than the use of text alone — would it perhaps be better to hide the mathematics altogether and represent the same information using a diagrammatic technique?

We explored quite a number of diagrammatic techniques, both OO and traditional in nature, looking for one which would enable us to present the meaning contained in a formal specification in a graphical format which could be more easily understood by user acceptors (and therefore be more successfully validated). It quickly became clear, however, that trying to represent the detail of a formal specification using a formal or semi-formal diagrammatic technique resulted in diagrams of exploding complexity — an infeasible approach.

In addition, even if the exploding complexity problem could be resolved, the use of two formal representations (the specification and a diagrammatic counterpart) would require that compatibility be maintained. Instead of trying to use two formalisms to represent the same information, we decided that it was more appropriate to use the formal specification to document the system requirements precisely, then use an informal diagrammatic technique to help illustrate the specification.

Given the decision to retain the Object-Z specification in the validation process, we re-examined OO diagramming techniques, looking for one which could be used to illustrate the meaning held in the Object-Z specification. The car example given earlier shows why traditional OO approaches are not adequate for non-development personnel — what was needed was a way of presenting the behaviour of a system, a ``roadmap'' to the formal specification.

We concentrated therefore on dynamic OO techniques. Unfortunately, of the OO approaches which include some form of dynamic modelling, most make no attempt to model the behaviour of the entire system, but rather provide piecemeal modelling using techniques such as state transition diagrams for each object [12]. This focus on modelling the behaviour of components of the system, rather than the behaviour of the system as a whole means that system-level behaviour cannot easily be conveyed.

We also considered non-OO approaches which held promise, in particular Petri nets [23,24] and Statecharts [17], both of which have OO extensions: OPNets [20] and ObjectCharts [7]. These notations proved to be unsuitable for the same reason that many of the notations we examined were unsuitable: they focus on formalising the properties of a system in some detailed manner (ultimately, one which is useful to system constructors). Consequently, when dealing with complex systems they become too complicated and inflexible as explanatory mechanisms. We required, not a second formalism to represent a system's behaviour,

but a less formal, flexible notation which could be used as a roadmap to guide a user acceptor through the formal definition of the system contained in the Object -Z specification.

A set of notations which seemed promising were those OO notations which do offer a way of structuring the presentation of the system from the user's point of view. These include "scenarios" [25] and "use cases" [19], which are also used by Lorenz [21], Booch [2] and Henderson-Sellers and Edwards [18]. Briefly, use cases and scenarios describe a behaviourally related sequence of transactions which a user of the system (either a human or another system) performs in dialogue with the system. Our results had supported the assertion made by the various authors of these notations: that describing the behaviour of a system in this manner facilitates a better understanding of the system by its user acceptors, permitting a more successful validation of the system requirements.

Unfortunately, although the use case/scenario approach seemed to be advantageous, these particular implementations were inadequate for our needs. In each case the notation was either non-graphical (or minimally so), making understanding on the part of the user acceptors more difficult, or they were too simplistic to enable the successful explanation of complex chains of related behaviour. We required a notation which would:

- describe the behaviour of the system from the point of view of a user of the system, whether that user be a human or another system. The better the understanding of the system specification by its users, the more successful the validation of requirements will be. A dynamic representation technique was required
- be simple but expressive — this point is related to the one above. A second formal notation was not required — the formality is captured by the Object-Z; the aim is more clearly to convey behaviour of the system to the users. Given this requirement, some kind of graphical representation was desirable
- support and describe an OO specification
- be able to handle specifications of any complexity. Some sort of hierarchical decomposition or abstraction mechanism was therefore required.

Ultimately we developed a notation which fulfilled these requirements[3]. The exact nature of the notation evolved during the development of the final specification prepared for DoSS — that of a corporate EDI gateway.

6 Conclusion — The Resulting Specification Structure

The last specification prepared for DoSS was of a corporate EDI gateway. DoSS had decided to start afresh with a redevelopment of SupplyNet and the redeveloper was to be selected following a tender process. One of the requirements of the tender was the development of a corporate EDI gateway for the whole of the State Government's EDI traffic. As very little material was (or still is) available on the requirements of such a gateway, the development of a formal specification was considered useful — both to DoSS and the organisations responding to the tender.

The notation we developed to assist in presenting system behaviours (called Event Chain notation), evolved iteratively over a period of months, as did the specification, which proved ultimately to be one of the most complex specifications we have produced.

The ultimate result of the evolutionary process associated with the development of the notation was a recommended specification structure for use with FOOM. We have found that the requirements specification needs to be broken into several sections, each with a different focus, to cater for the needs of different audiences.

6.1 The FOOM Specification Document

The various diagrammatic forms (products of the assorted types of modelling which occur) associated with FOOM each have a specific place within the specification document, which has a well defined structure:

[3] For a description of the notation see [14] and [34].

Tier One contains an informal description of the system designed for management and user participants without mathematical or systems development expertise, who are not involved in the ultimate acceptance of the specification developed (but who may be involved in approving the continued development of the system). This may be thought of as an executive overview, and describes the business setting of the system (its context), rather than the more abstract (and detailed) representation contained in section two. This section does not contain any formal statements

Tier Two contains the formal specification presented in terms of the event chain notation, and is designed to facilitate the validation of the requirements elicited via the user acceptors. Appropriate parts of the Object-Z specification (often, relevant fragments of class definitions drawn from the complete specification contained in the third tier of the FOOM document) are presented as required, using the event chain notation, to show how the specification formally models the behaviour of the system. The textual explanation accompanying the Object-Z fragments is presented in terms of the system setting and context — it is the system which is being explained, not the meaning of the mathematical statements. For the analyst/specifiers capable of understanding the mathematical formalisms the Object-Z is relatively straightforward, so there is no need to belabour the obvious. It is more important to convey the information not carried with the math — the context in which the system sits. The focus therefore is very much on behaviour and meaning, the contextual information which allows one to understand the system. For this reason the tier is also useful to developers, as it gives an entry point into what may be a large, complicated specification. For user acceptors who (realistically) probably won't have total confidence in their Z skills (a different issue from whether they can understand what is explained to them) the walkthrough is what allows them to understand the formal model well enough to validate it properly. Note that the same class schemata or operations may well be duplicated across various event chains, or even within the same one. This redundancy can be managed through the use of a supporting CASE tool.

Tier Three contains the complete Object-Z specification, documented in the traditional way (textual explanation surrounding the mathematics), complete with associated static MOSES diagrams and the communication model developed by Wafula and Swatman [34] to supplement the textual explanation. This section is suitable for systems designers and implementors already familiar with Object-Z, who may also gain benefit from first reading tiers one and two).

Each tier provides a view of the system useful to a particular class of audience. Providing a group of views, rather than a single all-purpose explanation, is required because the people with an interest in the specification have a variety of expectations, background, technical skills and understanding of the system. Examples of complete FOOM specifications may be found in [13].

Our results suggest that the approach is both viable from the point of view of validation and also feasible in a commercial IS setting (although the practicability of the approach has not been addressed in this paper). We believe that the validity of the approach is independent of the use of Z or Object-Z as an underlying specification language, a belief which has been supported by responses from the OO community. The notation is effective because the diagrams produced are simple. Rather than attempt to show everything on one diagram, which results in extremely complicated diagrams (the failing of most conventional OO diagramming techniques), the diagrams are simply an illustrative “roadmap” to the underlying OO specification; they show only the detail necessary to illustrate the behaviour of the system. This approach is not limited to the particular formalisms we have adopted: we believe this principle for structuring and presenting specifications would work for any formal language.

Thus far our research has demonstrated that the FOOM approach can be used in typical IS environments; in current work we are investigating:

- developing tool support for the approach, which will enable us better to apply the approach to real-world problems in a real- time framework (so far the approach has been applied to the development of specifications which were not particularly time critical) via a series of case studies
- undertaking comparative work, to assess the approach relative to those of others.

References

1 Boehm, B. (1976) Software engineering. IEEE Transactions on Computers, C-25(12), December, pp 1226-1241.

2 Booch, G. (1994) Object Oriented Analysis and Design with Applications, Benjamin Cummings, Redwood City, California, 2nd edition.

3 Brooks, F. (1987) No silver bullet: Essence and accidents of software engineering, Computer, pp. 10–18.

4 Checkland, P.B. (1981) Systems Thinking, Systems Practice, Wiley, Chichester.

5 Checkland, P.B. (1995). Soft systems methodology and its relevance to the development of information systems. In Stowell, F. A. (1995) Information Systems Provision: the Contribution of Soft Systems Methodology, McGraw-Hill Book Company Europe, Berkshire, England.

6 Checkland, P. & Scholes, J. (1989) Soft Systems Methodology in Practice, Wil ey, Chichester.

7 Coleman, D., Hayes, F. And Bear, S. (1992) Introducing objectcharts or how to use statecharts in object oriented design, IEEE Transactions on Software Engineering, 18(1), 9-18.

8 Cunningham, R.J, Finkelstein, A., Goldsack, S., Maibaum, T. And Potts, C. (1985) Formal requirements specification—the Forest project, Proceedings of the International Workshop on Software Specification and Design, London, 186-191.

9 Davis, A. (1994) Software Requirements: Objects, Functions and States, Prentice Hall, Englewood Cliffs.

10 DeMarco, T. (1979) Structured Analysis and System Specification, Yourdon press, New York.

11 Duke, R., King, P., Rose, G. & Smith, G. (1991) The Object-Z specification language, Technical Report 91-1, Software Verification Research Centre, Department of Computer Science, University of Queensland, Australia.

12 Fichman, G. And Kemerer, C. (1992) Object-oriented and conventional analysis and design methodologies, Computer, 22-39.

13 Fowler, D.C (1996) Formal Methods in a Commercial Information Systems Setting: the FOOM Method, PhD Thesis, Swinburne University, Melbourne.

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

15 Fraser, M., Kumar, K. & Vaishnavi, V. (1994) Strategies for incorporating formal specifications in software development, Communications of the ACM, 37(10), 74 –86.

16 Hall, A. (1990) Seven myths of formal methods, IEEE Software, 7(5), 11-19.

17 Harel, D. (1987) Statecharts: a visual formalism for complex systems, Science of Computer Programming, 8, 231-273.

18 Henderson-Sellers, B. And Edwards, J.M. (1994) BOOKTWO of Object Oriented Knowledge: The Working Object, Prentice Hall, Sydney.

19 Jacobson, I., Christerson, M., Jonsson, P. And Overgaard, G. (1992) Object Oriented Software Engineering: A Use Case Driven Approach, Addison-Wesley, Wokingham, England.

20 Lee, Y. And Park, S. (1993) Opnets: an object oriented high-level petri net model for real-time system modelling, Journal of Systems Software, 20, 69-86.

21 Lorenz, M. (1993) Object Oriented Software Development, Prentice Hall, Englewood Cliffs

22 McMorran, M. & Powell, S. (1993) Z Guide for Beginners, Blackwell Scientific Publications, Oxford.

23 Peterson, J. (1977) Petri nets, ACM Computing Surveys, 9(3), 223-252.

24 Peterson, J. (1981) Petri Net Theory and the Modelling of Systems, Prentice Hall, Englewood Cliffs.

25 Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. And Lorenson, W. (1991) Object Oriented Modelling and Design, Prentice Hall, Englewood Cliffs.

26 Siddiqi, J. (1994) Challenging universal truths of requirements engineering, IEEE Software, 11(2), 18–19.

27 Sommerville, I. (1989) Software Engineering, Addison-Wesley, Wokingham, third edition.

28 Susman, G.I. and Evered, R.D. (1978) An assessment of the scientific merit of action research, Administrative Science Quarterly, 23, 582-603.

29 Swatman, P.A. (1993). Using formal specifications in the acquisition of information systems: Educating information systems professionals. In J.P. Bowen & J.E. Nicholls, editors, Z User Workshop: London 1992, Workshops in Computing, pages 205–239. Springer Verlag, London.

30 Swatman, P.A. & Swatman, P.M.C. (1992). Formal specification: An analytic tool for (management) information systems. Journal of Information Systems, 2(2), 121–160.

31 Swatman, P.A. & Swatman, P.M.C. (1992). Managing the formal specification of information systems. Proceedings of the International Conference on Organization and Information Systems, Bled, Slovenia.

32 Swatman, P.A., Swatman, P.M.C., & Duke, R. (1991). Electronic Data Interchange: A high-level formal specification in Object-Z. In P.A. Bailes, editor, 6th Australian Software Engineering Conference (ASWEC'91): Engineering Safe Software, Sydney, NSW.

33 Swatman, P.A., Fowler, D.C., & Gan, C.Y.M. (1992). Extending the useful application domain for formal methods. In J.E. Nicholls, editor, Z User Workshop: York 1991, Workshops in Computing. Springer Verlag, London.

34 Wafula, E.N. and Swatman, P.A. (1996) FOOM: A Diagrammatic Illustration of Object-Z Specifications, Object Oriented Systems, 3(4), December.

35 Williams, L. (1994) Assessment of safety-critical specifications, IEEE Software, 11(1), 51–60.

36 Wordsworth, J. B. (1993) Software Development with Z: A Practical Approach to Formal Methods in Software Engineering, Addison-Wesley.

37 Wing, J.M. (1990) A specifier's introduction to formal methods, IEEE Computer, 23(9), 8-24.