ELECTRONIC WORKSHOPS IN COMPUTING

Series edited by Professor C.J. van Rijsbergen

Antony Bryant and Lesley Semmens (Eds)

Methods Integration

Proceedings of the Methods Integration Workshop, Leeds, 25-26 March 1996

Paper:

Improving Access of the Commercial Software Developer to Formal Methods: Integrating MERISE with Z

Lynne Dunckley and Andrew Smith

Published in collaboration with the
British Computer Society

Improving Access of the Commercial Software Developer to Formal Methods: Integrating MERISE with Z

Lynne Dunckley

Department of Computing ,University of Luton,
Luton, Bedfordshire.

Andy Smith

Department of Computing ,University of Luton,
Luton, Bedfordshire.

Abstract

Whilst formal methods can make valuable contributions to system specification, structured methodologies are seen as more appropriate for commercial systems development, particularly user involvement. This paper investigates the applicability of Z to MERISE, the French structured methodology, recommending the adoption of a formal link between the two, an integration procedure proposed by Semmens *et al.*

We apply our previous classification of approaches to user-centred design; *participative structures*, *processes*, and *scope* which need to be addressed by a variety of methods, techniques and tools. We seek to show how combination with structured methods can improve the accessibility of formal methods, *processes* can be enhanced, enabling better communication through *participative structures*.. These ideas are illustrated by means of a simple, albeit real-life example in which a Z specification is developed through the translation of MERISE conceptual data and process models.

1 Introduction

The general term software crisis is used to refer to the potentially intractable problems associated with software development. Over the last decade, in an attempt to find solutions to the software crisis, more and more software development has involved methods which are both founded within a sound engineering philosophy and are accessible by commercial software developers and their users. A key mechanism for providing the engineering background has been the use of formal methods, and Z is one of the most widely used methods. A significant element in accessibility is the user centredness of the overall approach.

1.1 Survey of Systems Design

During 1992/3 a major survey [20] of UK commercial organisations was undertaken in order to elicit approaches to systems design and implementation. Responses were received from the IT Manager, or equivalent, of 66 medium to large user organisations. From the survey it emerged that whilst there is considerable evidence that traditional (pre-structured) design methods still dominate the design process, there is a wide variety of other methodologies in use. Structured Methods were shown to be in use in a large number of organisations and SSADM was the method most quoted. The shift to structured methods was shown to have come about mainly for reasons of efficiency and effectiveness within the design process.

Respondents were asked to indicate the tools used to communicate and specify their design within the design team (internal) and with the user community (external). A range of options was provided from soft to formal techniques. The results are shown in Table 1. It was quite surprising to note that a considerable number of respondents used some form of formal method at some stage within their selected project. Over half the organisations reported the use of such methods at some stage in systems development. The authors feel that a rather loose interpretation of the term formal was assumed by the respondents, but the results are nonetheless significant. Further research is required to ascertain the specific formal method and detailed application area.

1.2 User Centred Design (UCD)

Effective user engagement is now generally perceived to be a vital ingredient in the successful take-up of Information Technology projects. In practice there seems to be a lack of clarity between academics and practitioners about how user engagement can be enhanced and developed into a fully user centred approach. Whilst the underpinning elements of UCD have been well explored [11] and levels of user involvement identified [9,10] there has been only limited success in modifying current commercial practice to enhance user centredness. In order to fully analyse user involvement the authors [20] have shown how practical approaches to user-centredness in design can be evaluated under the following broad concepts: *participative structures, processes, and scope*.

Table 1
Survey of approaches to systems design and implementation
Tools used during the project to specify and communicate system design
(average of internal and external data)

Tool	Percentage of organisations
Flow Diagram based	67.8
Formal Specification	63.7
Natural Language	33.4
Rich Pictures.	9.9
Object Oriented Methods	5.3
Other / in-house	14.4

Participative Structures

Structures to support the design process include project management, design team structures, and mechanisms for user involvement and communication between the design team and the user community.

Processes

Processes for design focus around the design methodology and include issues relating to requirement elicitation, communication and implementation, including approaches to prototyping.

Scope

The scope of the design process relates to how far the analysis reflects a socio-technical, as opposed to just a technical, solution. More specifically an IT project with only a narrow scope will focus on technical and functional requirements whereas one with a wider scope will address issues such as the allocation of function between man and machine, the design of work structures and individual jobs and ways of enhancing job satisfaction within the organisation.

For effective and successful IT systems design and implementation all three of these concepts need to be adequately covered. A variety of methods, techniques and tools need to be used which can address all these issues. Compatibility and transferability between techniques is crucial. In this paper we seek to show how the accessibility of formal methods *processes* can be enhanced thereby enabling better communication through user centred *participative structures*.

1.3 Integrating Formal and Structured Methods

As formal methods have been extended into new application areas, a number of developments have been suggested. One such recent divergence or development from basic Z has been to embed Z notation into structured systems methodologies. As the use of formal methods increases such development will become more significant. It will, however, be important to retain the essential user-centredness of the structured approach.

A structured methodology is an approach to software development which breaks a system down into component parts so that data and processes can be treated separately. Such methods have the advantage that while they impose a discipline on software development, the *processes* adopted and the diagrams produced can be understood intuitively by the non-expert user and therefore facilitate communication between the user and developer. Formal methods, on the other hand, are not generally seen as an appropriate mechanism for facilitating user communication and tend to be lacking in terms of *participative structure* and *scope* needed to develop successful systems. They are, of course, unambiguous and can be reasoned about. This paper considers an approach which would combine the precision of formal methods with the simplicity and clarity of diagrams found in structured systems methodology. The objective of combining requirements capture and structuring techniques with formal methods is to facilitate mathematical proof such as would be required for verification of mission and safety critical systems. It also offers the advantages of using formal methods for key areas of the specification and structured tools for those parts of the development unsuitable for formal methods e.g. business analysis, strategic issues, initial modelling and design.

A number of researchers have investigated the combination of formal and structured methods. Bradley [2] for example reported the results of applying the combined method to a number of specifications in the avionics sphere. Clarke [4] also described how formal methods can be incorporated into the software development life-cycle. Semmens and Allen [16] have described a method which combines Yourdon's structured design method and Z. Whilst a number of researchers have investigated the integration of Z with SSADM, including Draper [7] who described an industrial approach in the development of secure systems. Polack, Whiston and Hitchcock [11] have reported the development of a method for writing Z specifications using SSADM version 4. Polack and Mander [13] have described the benefits of combining SSADM with Z to produce the SAZ method. Swatman and Swatman [19] have investigated the application of Z and Object Z within the domain of Management Information Systems.

Semmens *et al* [17] have categorised the main approaches as:

- (i) the use of both structured and formal methods side by side without formal links
- (ii) the adoption of a formal link between the two methods
- (iii) the use of structured analysis as the front end to the formal method.

For reasons which will become clear this paper analyses the applicability of MERISE to Z and recommends an integration procedure corresponding to Semmens's second option.

2 Merise and Z

2.1 Selecting MERISE

In selecting a structured methodology to combine with a formal method it is essential that the underlying philosophy of both methods should be compatible and that the view of data and processes should be consistent throughout the integrated method. This paper suggests MERISE as an appropriate methodology and explores the process of combining Z and MERISE in a way which will overcome the weaknesses of each separate approach. MERISE is one of the most widely used methodologies, originating in France in 1979 but with its influence subsequently spreading to Spain, Switzerland, and North America [3,21]. It may form the basis of Euromethod.

Specifically MERISE has been chosen for the following reasons:

- (i) the method is quite wide in *scope* and addresses several aspects of systems development not addressed by SSADM / Yourdon which tend to focus on technical and functional aspects and do not include business and organisational objectives;
- (ii) the method uses a number of *processes* which take account of both the static and dynamic aspects of an information system;
- (iii) it is often used in conjunction with CASE tools;
- (iv) integration methods such as SAZ could easily be adapted to operate with the method;
- (v) it can be used to model concurrent systems in a way not readily achieved by SSADM.

2.2 Introduction to MERISE

MERISE is based on the philosophy that the business enterprise is a group of dynamically interacting elements organised according to an objective, and within an environment. The enterprise is modelled as consisting of three interacting sub-systems; an operations system, an information system and a guidance system, with each interacting information. There is also a recognition within this philosophy that there will be formal and informal information systems. The formal information system is identified by codified rules of operation whilst the informal system is based upon relationships between individuals and work groups. It is clear, therefore, that MERISE offers much to developers seeking a user-centred approach and / or a wide scope in design.

MERISE consists of six stages: strategic planning, preliminary study, detailed study, development, implementation, maintenance. Within the detailed study analysis is based on an abstraction cycle, which aims to design an information system by following a modelling logic consisting of three levels:

<i>conceptual</i>	<i>what are we trying to do?</i>
<i>logical / organisational</i>	<i>who, when and how?</i>
<i>physical</i>	<i>what means and constraints</i>

The purpose of the conceptual level is to capture the management rules which underlie the business. This conceptual level uses a conceptual data model (CDM) to represent all the data used by a firm and a conceptual process model (CPM) to describe the activities of the firm. The data models and the processing models are developed independently, even to the extent that different analysts may be involved. Furthermore, as discussed by Avison [1], unlike SSADM, although the two types of models are compared, there is no attempt to link the process models and the data models or cross-reference them for consistency.

In this paper the authors are proposing to integrate MERISE with Z through the use of specific links at certain stages within the development life cycle (category (ii) in Semmens's classification), particularly within the detailed design stage. The integration of MERISE with a formal method would allow the examination of the detail of the data and process models to verify the consistency of both. MERISE uses a data modelling process which identifies the entities involved in the system, the relationship between these and the functional integrity constraints linked to each relationship. The conceptual process model, regards a process as a set of operations defined from the point of view of the management rules of the firm [14].

The purpose of this paper is to investigate the application of a combination of the MERISE methodology and formal methods to a small application by refining the conceptual models produced by the systems methodology using Z. The authors are keen to apply the approach to a full-scale application, probably derived from one of the survey respondents. A preliminary study has, however, been undertaken within a separate organisation with which the authors have enjoyed contact.

3 Application - Preliminary Study

3.1 The Problem

The organisation used in this preliminary study provides gas appliances to builders for show houses as part of their marketing strategy. Within the organisation there is a housing development department whose responsibility it is to attract new customers and promote the spread of gas to villages and new estates. Actual sales of appliances are handled by customer services. The finance department deals with billing customers and accounts. The goods are supplied from stocks at the regional office headquarters. Unfortunately at the time of initial contact, the system under which the loan of appliances to show houses was made, had become subject to unacceptable losses. In one year five appliances have been returned in a damaged state and many were missing altogether.

3.1.1 Outline System

At the time of the initial study the system was described as follows:

'The builder requests the loan of appliances from the housing development department. A representative of this department sorts out the details of the appliances and the conditions of the loan. No date is fixed for the termination of the loan. A letter is sent to the builder confirming the arrangements and detailing the conditions. The representative forwards a loan voucher to the customer services department. This contains the builder's name, show house address and list of appliances. A representative from the housing development department periodically visits the show house to check the appliances and to ascertain whether the house has been sold. The builder is supposed to notify the housing development representative when the appliances are no longer required or if he wishes the appliance moved to another house.'

3.2 MERISE Analysis

3.2.1 Initial Stages

One of the strengths of MERISE is that it combines an examination of business objectives and critical success factors, during the initial stages with a technical design process. In this case the business objectives of the system would be identified and it would be recognised that the system was really concerned with introducing gas sales by promoting gas appliances in new homes: The immediate objective being to minimise the cost of this promotion.

3.2.2 The Conceptual Data Model

The second approach of MERISE, the conceptual model level, represents the information system independently of the business organisation. The objective is to capture the rules of the system and represent these in a conceptual data model. The MERISE conceptual data model will firstly identify the main entities in the system and document the attributes associated with them. In MERISE the basic components of the data model are called entities (or objects). An entity has a unique identifier or code and one or more properties which distinguish the occurrences of the entity. Multi-valued properties are not allowed. An entity is represented graphically by a rectangular shape with a description in the top. For example in this system the entities are as shown in Figure 1.

A relationship in MERISE is a semantic link between one or more entities. A relationship, which is represented in the diagram as an ellipse containing a brief description. The cardinalities of an entity in linking in a relationship indicate the minimum and maximum number of occurrences of the relationship to which each entity is linked. A relationship may also have a number of properties associated with it. The conceptual model is developed gradually from the preliminary study stage through to the design stage. The graphical representation of the conceptual data model is based on the entity-attribute-relationship model proposed by Chen [6]. Recent developments of MERISE apply ER modelling using individual formalism which is semantically enriched[5,15].

Figure 1 : Entities

An outline of the conceptual data model for the preliminary system is shown in Figure 2, where the cardinalities are shown for each relationship between the entities. In this application, since entities in MERISE may not have multi-valued attributes, in addition to *LOANVOUCHER*, *LVLINE* (loan voucher line) is needed to describe each appliance covered by the loan agreement.

Two pairs of cardinalities (a minimum and a maximum value) are shown for each relationship, one pair attached to each entity. For example, builder and showhouse are represented with *SHOWHOUSE* (1,1), the cardinality indicating that one showhouse belongs to only one builder, but *BUILDER* (0,N) indicates that a builder can have between zero and many showhouses.

3.2.3 Functional Integrity Constraints

In MERISE the concept of the functional integrity constraints (FIC) is added to the conceptual data model. The FIC is used to indicate that one of the entities linked in a relationship is determined by a knowledge of the others. In the case of binary relationships a FIC is indicated by the cardinality (1,1). In Fig 2, for example :

- FIC 1. A showhouse is built by only one builder.
- FIC 2. A builder is dealt with by only one representative.
- FIC 3. A loan voucher is issued by one representative.
- FIC 4. An appliance is checked by only one representative.
- FIC 5. An appliance is located in one house only.

It is common practice not to add the FIC notation to the diagram in the case of (1,1) cardinality and in the case of Fig 2 other FIC's are identified by the cardinality (1,1) from APPLIANCE to LVLIN, and from LOANVOUCHER to BUILDER.

The CDM is created from any existing data dictionary and input and output documents. Errors can arise from the lack of clear definition and the occurrence of polysemes and synonyms. These need to be detected and the semantic rules clearly defined. The usual method employed is rather *ad hoc* and relies on cross checking between analysts and users. Errors can go undetected even when CASE tools are used. The method proposed in this paper is to translate the CDM into a Z specification at the design stage in order to introduce greater rigor, and to check the CDM for completeness and consistency, provided by the ability to reason about the Z specification.

Figure 2 : Conceptual Data Model

3.3 Abstract Design in Z

3.3.1 Conceptual Data Model Transformation

This section gives an illustration of the style in which a Z specification can be developed from the conceptual data model. The translation procedure would involve:

- 1) Representing each entity from the CDM by a schema which describes the main properties of the entity and must contain an unique code to act as an identifier for each occurrence ;
- 2) Describing each relationship between entities including, when required, properties of the relationship.
- 3) Adding schema details for cardinalities and functional integrity constraints.
- 4) Summarising the CDM in a state invariant schema which describes the entity relationships in the signature and adds constraints corresponding to the system boundary and the business rules;
- 5) Checking the specification for consistency and completeness. The use of a Z type checker here would be an advantage.

The translation into Z must take into account the key differences between the data model used in the CDM and, say the LDS in SSADM., especially the precise use made of cardinalities and functional integrity constraints.

Another difference in data modelling in MERISE is that the conceptual model could include an n-ary relationship. This situation often arises in models derived from E-R diagrams where a child entity can have two parents and is determined by them.

Although an n-ary relationship is not featured in the application CDM we have illustrated the concept by translating the example shown in Fig 3 which is taken from Quang and Chartier-Kastler [14].

Figure 3 : n- ary Relationship

This n-ary relationship can be described in the following schema.

<i>Parentage</i>	
x	: <i>MOTHER</i>
y	: <i>FATHER</i>
z	: <i>CHILD</i>
<i>motheredby</i>	: <i>CHILD</i> $\rightarrow$ <i>MOTHER</i>
<i>fatheredby</i>	: <i>CHILD</i> $\rightarrow$ <i>FATHER</i>
<i>childof</i>	: <i>CHILD</i> $\rightarrow$ (<i>MOTHER</i> $\times$ <i>FATHER</i>)
$\forall z : CHILD ; \exists x : MOTHER ; \exists y : FATHER \bullet (z, x) \in motheredby$ $\wedge (z, y) \in fatheredby \Leftrightarrow childof(z) = (x, y)$	

3.3.2 Z Specification of CDM

The first stage in the specification would be to declare the basic types to be used in the remainder of the specification. The entities represented on the CDM belong to the given sets for the specification, i.e. for the entities:

[*APPLIANCE, BUILDER, HOUSE, LOANVOUCHER, LVLIN, PERSON*]

PERSON is the set of all people, some of whom may be representatives of the housing development department. *BUILDER* is the set of all builders, *APPLIANCE* and *HOUSE* are also sets of all possible objects of their type. Where attributes have been identified in detail the entity would be declared as a schema, as for example *APPLIANCE*, after the declaration of the attribute types as given type :

[*SERIALNO*, *MAKE*, *MODEL*, *DESCRIPTION*]

The use of Z free types for attributes like *DESCRIPTION* can add to the power and clarity of the specification, e.g.

<i>APPLIANCE</i>	
<i>serialno</i>	: <i>SERIALNO</i>
<i>make</i>	: <i>MAKE</i>
<i>model</i>	: <i>MODEL</i>
<i>desc</i>	: <i>DESCRIPTION</i>

In the same way entity relationships would be described, for example by a schema *Rel_records* between *LVLINE* and *LOANVOUCHER* which can be represented by both a relation and its inverse, a total surjective function, so that every *LOANVOUCHER* is included in a mapping from *LVLINE*. Constraints ensure the correct cardinalities and that a *LVLINE* cannot exist without its *LOANVOUCHER*. In a database implementation this would enforce the use of integrity constraints such as 'delete cascades'.

<i>Rel_records</i>	
<i>Lvoucher_id</i>	: <i>LOANVOUCHER</i>
<i>Lvline_id</i>	: <i>LVLINE</i>
<i>record</i>	: <i>LVLINE</i> $\rightarrow$ <i>LOANVOUCHER</i>
<i>comprise</i>	: <i>LOANVOUCHER</i> $\leftrightarrow$ <i>LVLINE</i>
<i>record</i> $\sim$ <i>comprise</i>	
ran <i>comprise</i> = dom <i>record</i>	
# <i>comprise</i> ($\{ \{Lvoucher_id\} \}$) ≥ 1	

In the same way the relationship between *HOUSE* and *BUILDER* would be described by the following schema.

<i>Rel_builds</i>	
<i>name</i>	: <i>BUILDER</i>
<i>address</i>	: <i>HOUSE</i>
<i>built</i>	: <i>HOUSE</i> $\rightarrow$ <i>BUILDER</i>
<i>owns</i>	: <i>BUILDER</i> $\leftrightarrow$ <i>HOUSE</i>
<i>built</i> $\sim$ <i>owns</i>	
ran <i>owns</i> = dom <i>built</i>	
# <i>owns</i> ($\{ \{name\} \}$) ≥ 0	

In general it is always better to develop specifications with functions replacing relations as often as possible. The inverse of most one-to-many relations will be a function. In this application this can be summarised as the following functional relationships :

- A show house is built by one builder.
- An appliance is located in one house at a time.
- A loan is approved by one representative.
- An appliance is issued to one builder at a time.

The FICs shown on the MERISE conceptual model (Fig 2) are named as *built* for FIC1, *loan* for FIC2, *issue* for FIC3 and *check* for FIC4. The predicates which constrain the specification can be expressed as :

A builder must have a house in order to get a loan.
 An appliance is still part of stock.
 Every appliance is checked, only representatives can make loans.

The next stage is to create the Z schema for the state invariant properties of the Appliance loan system. In the signature (first section) the entities and functional integrity constraints are described, whilst the constraints describe the system boundaries. Partial functions are used when subsets of the given sets are involved within the system and total functions when the given sets are completely encompassed by the system.

This includes the entities, relationships and most of the FICs from the conceptual data model and is given in the schema below :

<i>Appliance _ loan</i>	
<i>stock</i>	: $\mathbb{P}APPLIANCE$
<i>company</i>	: $\mathbb{P}BUILDER$
<i>show_house</i>	: $\mathbb{P}HOUSE$
<i>representative</i>	: $\mathbb{P}PERSON$
<i>built</i>	: $HOUSE \rightarrow BUILDER$
<i>locate</i>	: $APPLIANCE \nrightarrow HOUSE$
<i>check</i>	: $APPLIANCE \nrightarrow PERSON$
<i>loan</i>	: $BUILDER \nrightarrow PERSON$
<i>issue</i>	: $LOANVOUCHER \rightarrow PERSON$
<i>record</i>	: $LVLINE \rightarrow LOANVOUCHER$
<i>cover</i>	: $LVLINE \rightarrow APPLIANCE$
<hr/>	
$\text{dom } loan \subseteq \text{ran } built$	
$\text{dom } locate \subseteq stock$	
$\text{dom } built \subseteq show_house$	
$\text{ran } built \subseteq company$	
$\text{ran } loan = representative$	
$\text{ran } cover = \text{dom } locate$	
$\text{dom } locate = \text{dom } check$	
$\text{ran } check = \text{ran } record \circ issue$	

The initial state of the loan system is described so that initially no loans exist whilst stock exists independently in other parts of the system.

<i>Init</i>	
<i>Appliance_loan</i>	
$loan' = \emptyset$	
$issue' = \emptyset$	
$stock' \neq \emptyset$	
$company' = \emptyset$	

3.3.3 The Conceptual Process Model

MERISE does not use diagrams to model the flow of information through at the conceptual level although these diagrams are often used in the operational processing model at a later stage in the life cycle. Instead the conceptual process model is developed in parallel to the CDM usually by a different group of people and the two models are intended to be compared but not formally checked against each other. The process model consists of processes, which are sets of operations, events and synchronisations. An event is an activity which, on its own or when synchronised with other events, can trigger an operation within the system. An event can also result from an operation and is then termed a result. An operation must satisfy certain preconditions and depends on activating a synchronisation. The process is represented graphically in Figure 4 as :

Figure 4 : Conceptual Process Model

An issuing rule at the output stage of an operation is used to decide which results are to be triggered according to the events of an operation e.g.

- OK - if result is acceptable
- NOT OK - if result is unacceptable
- NR - if nothing to report.

Using these concepts, part of the conceptual processing model relating to the operation for approving a new builder, for the application, is shown in Figure 5.

The authors propose that the CPM should also be verified by translation into Z but clearly this cannot be done until after at least some of the translation of the CDM has been carried out. The CPM can be seen to be conceptually nearer to the Entity Life Histories used in SSADM than to data flow diagrams. However the CPM is developed at a much earlier stage in the design than the ELH in SSADM. It is of note that Polack and Mander [12] have advocated the use of ELH as a key stage in the SAZ method.

The following schemas illustrate the way in which the operations from the conceptual process model can be translated into Z schemas. For example, the next schema describes the operation 'New_Builder_Approval' from the conceptual process model, given in Figure 5.

The predicates of any new schema will specify those values which satisfy the state invariant and for which there exists final states satisfying both the initial and final states. In the example below, *Approve_Builder*, the builder must not already be approved and must not have had existing loans. For the sake of brevity constraints for the unchanged primed components of Δ *Appliance_loan* have been omitted from the schema [18].

Figure 5: Part of Conceptual Process Model

Although this corresponds to the main operation for 'New Builder Approval' in Figure 5 from the CPM, it is not sufficient to cover the results of the operation. To deal with this we must define an enumerated type :

RESPONSE ::= *OK* | *NOT OK* | *Nothing to Report*

The operation needs to cover the 'Not OK' result when the builder is not approved. The schema *NotOK_Builder*, uses the *RESPONSE* type to describe this :

<i>NotOK_Builder</i>	
$\approx$ <i>Appliance_loan</i>	
<i>name ?</i> : <i>BUILDER</i>	
<i>reply !</i> : <i>RESPONSE</i>	
<hr/>	
<i>name ?</i> $\in$ <i>company</i>	
<i>reply!</i> = <i>NOT OK</i>	

In addition the 'OK Builder' response requires a further small schema to represent the state when the builder is approved.

$$OKMessage \hat{=} [\textit{reply !} : \textit{RESPONSE} \mid \textit{reply!} = \textit{OK}]$$

The whole '*New_Builder_Approval*' operation can now be described by a combination of the three schema :

$$\textit{New_Builder_Approval} \hat{=} (\textit{Approve_Builder} \wedge \textit{OKMessage}) \vee \textit{NotOKBuilder}$$

The same procedure can be used to specify the 'Check In Stock' operation from Fig 5. The '*In_Stock*' schema checks whether an appliance is in stock. The schema sets up the preconditions for making a loan, i.e. the appliance must be in stock, not already in a showhouse and not already loaned to a builder.

<i>In_Stock</i>	
$\approx$ <i>Appliance_loan</i>	
<i>appid ?</i> , <i>loan_appid!</i> : <i>APPLIANCE</i>	
<hr/>	
<i>appid ?</i> $\in$ <i>stock</i>	
<i>appid?</i> $\notin$ <i>dom locate</i>	
<i>appid?</i> $\notin$ <i>ran cover</i>	
<i>loan_appid!</i> = <i>appid ?</i>	

In the same way the result of an appliance being out of stock can be described in the *Out_of_Stock* schema.

<i>Out_of_Stock</i>	
$\approx$ <i>Appliance_loan</i>	
<i>appid?</i> : <i>APPLIANCE</i>	
<i>reply !</i> : <i>RESPONSE</i>	
<hr/>	
<i>appid ?</i> $\notin$ <i>stock</i>	
<i>reply!</i> = <i>NOT OK</i>	

The complete 'Check In Stock' operation can now be described with both results.

$$\textit{Check_In_Stock} \hat{=} (\textit{In_Stock} \wedge \textit{OKMessage}) \vee \textit{Out_of_Stock}$$

Finally the *Make_Loan* schema describes the main functions of 'Make Appliance Loan' operation in Fig 5.

The CPM dictates that the *Make_Loan* operation is preceded by a synchronisation of *In_Stock* and *Approve_Builder*. The schema *Make Appliance Loan* receives the outputs of *In_Stock* and *Approve_Builder* using the schema piping operator to provide the inputs to *Make_Loan* [8, 22].

$$\textit{Make Appliance Loan} \hat{=} (\textit{Approve_Builder} \wedge \textit{In_Stock}) >> \textit{Make_Loan}$$

<i>Make_Loan</i>	
Δ <i>Appliance_loan</i>	
<i>loan_name</i> ?	: <i>BUILDER</i>
<i>loan_appid</i> ?	: <i>APPLIANCE</i>
<i>address</i> ?	: <i>HOUSE</i>
<hr/>	
<i>address</i> ? $\in$ <i>show_house</i>	
<i>built</i> (<i>address</i> ?) = <i>loan_name</i> ?	
<i>locate</i> ' = <i>locate</i> $\oplus$ { <i>loan_appid</i> ? $\mapsto$ <i>address</i> ? }	

Unlike SSADM, MERISE does not preempt the selection of an implementation data model by selecting and utilising primary and foreign keys. The entities and attributes are required to have unique codes solely in order to distinguish between occurrences. In MERISE the CDM is transformed into a Logical Data Model which is suitable for computerisation. This is illustrated in the next operation shown in Fig 5 for the creation of the *LOAN_VOUCHER* and *LVL* (loan voucher lines).

<i>Create_Loan_Voucher</i>	
Δ <i>Appliance_Loan</i>	
<i>Lvoucher_id</i> ?	: <i>LOANVOUCHER</i>
<i>rep</i> ?	: <i>PERSON</i>
<hr/>	
<i>Lvoucher_id</i> ? $\notin$ <i>dom issue</i>	
<i>issue</i> ' = <i>issue</i> $\oplus$ { <i>Lvoucher_id</i> ? $\mapsto$ <i>rep</i> ? }	

The creation of *LVL* (loan voucher lines) can be specified without at this stage using the concept of foreign keys.

<i>Create_LVLine</i>	
Δ <i>Appliance_Loan</i>	
$\cong$ <i>Make_Loan</i>	
$\cong$ <i>Create_Loan_Voucher</i>	
<i>loan_appid</i> ?	: <i>APPLIANCE</i>
<i>reply</i> !	: <i>RESPONSE</i>
<i>Lvoucher_id</i> ?	: <i>LOANVOUCHER</i>
<i>LVL-id</i> ?	: <i>LVL</i>
<hr/>	
<i>LVL_id</i> ? $\notin$ <i>dom record</i>	
<i>LVL_id</i> ? $\notin$ <i>dom covers</i>	
<i>record</i> ' = <i>record</i> $\oplus$ { <i>LVL_id</i> ? $\mapsto$ <i>Lvoucher_id</i> ? }	
<i>cover</i> ' = <i>cover</i> $\oplus$ { <i>Lvoucher_id</i> ? $\mapsto$ <i>loan_appid</i> ? }	
<i>Lvoucher_id</i> ? $\in$ <i>dom issue</i>	
$\forall z : LVL ; \exists x : APPLIANCE ; \exists y : LOANVOUCHER \bullet$	
$(z,x) \in cover \wedge (z,y) \in record \iff y \in dom issue$	
<i>reply</i> ! = <i>OK</i>	

To summarise, the CPM is translated into Z at the end of the detailed study stage of MERISE. This would ensure consistency between the CPM and the CDM entity type and constraint rules. The translation would consist :

- For each main process in the CPM create a schema by converting triggering events into preconditions, using the relevant entity types from the CDM.
- Combine the schemas from each process using schema calculus to represent synchronisation requirements.
- Build the process operation from the tasks listed, consistent with the business rules captured in the state invariant.

- d) Create separate schemas for each outcome of the process with appropriate preconditions.
 - e) Combine the postconditions from the operation with the outcome schemas using schema calculus.
- In this example Z can be seen to be sufficiently powerful to express all the requirements of the conceptual models in MERISE.

4 Conclusions

The integration of a structured method with a formal method is possible provided that the fundamental concepts of both approaches are consistent. This paper has put forward a method for integrating MERISE with Z which corresponds to Semmens *et al* [17] second option by developing definite links between the two methods. This exploits the abstraction cycle of MERISE, transforming conceptual models into Z to give increased precision for validation and verification purposes.

In the case of MERISE and Z the fundamental concepts of both applications are consistent and the methodologies are shown to be complementary. The structured method provides the basic framework, whilst Z can be used to provide the detail, for example, the conceptual data model can be transformed relatively easily to a Z specification. The construction of the state invariant is facilitated by the identification of the entities, relationships and functional integrity constraints (FICs). In the same way the conceptual process model can be used to specify the operations described in the Z specification. The graphical representations provide the primary specification of the operations but where greater precision is needed the power of Z is available. By combining the strengths of both Z and MERISE the systems designer is able to 'get the best of both worlds' and to benefit from an overall increase in the power of the facilities available.

The formalities available in Z, including the facilities for formal proof can be translated and communicated to the user through the intuitive modelling techniques of MERISE. Through the adoption of an integrative Z / MERISE approach the accuracy of the specification and consequent quality of a range of software systems should be enhanced whilst maintaining the advantages that MERISE offers to the developer seeking a user-centred approach. In terms of the broad concepts of UCD the user-centredness of the process is enhanced and users are able to communicate with developers through participative structures much more effectively. The authors are seeking to test the method in a wider application context in order to demonstrate how the integration MERISE with Z can improve the access of the commercial software developer to the benefits afforded by formal methods.

Future work will involve prototyping with a larger application and investigating the integration of later stages of MERISE with Z. In particular the optimisation of the logical data model (LDM) which is located between the conceptual data model (CDM) and the physical data model (PDM) is an area where the definite link could be established between MERISE and a formal method, such as Z. The logical data model transforms the abstraction state of the CDM into one concrete state suitable for computerisation. For example, in practice this selection of the concrete state has often been between a relational and codasyl data model on which to base the LDM.

5. Acknowledgement

The writers would like to acknowledge the helpful discussion provided by colleagues D. Heath, and D. Allum .

6. References

1. Avison , D.E. MERISE: A European methodology for developing information systems. European Journal of Information Systems. 1991;Vol 1, p183.
2. Bradley, A. Embedding formal methods in SAFRA. Proc. Software engineering and its applications to avionics, 1988; pp 101-9, AGARD.
3. Banos, D and Malbosc, G. Merise pratique:les points clés de la methode; Editions Eyrolles, Paris. 1988.
4. Clarke, S. The increasing importance of formal methods (software engineering). Computer Bulletin. 1988; 4 pt.1 pp22-3.
5. Caron C., Bedard Y. Extending Individual Formalism for a more complete modeling of Urban Spatially referenced data. Comput. Environ, and Urban Systems. 1993; Vol 17 pp 337-346.
6. Chen, P. The entity-relationship model toward a unified view of data. ACM TODS, 1976; Vol 1, No 1 pp.9-36.

7. Draper, C Practical experiences of Z and SSADM. Z User Workshop. 1992;pp 9-36 Bowen, J.P., Nicholls, J.E. (ed). Oxford, Springer-Verlag.
8. Heath, D., Allum, D. and Dunckley, L Introductory Logic and Formal Methods. Alfred Waller. 1994.
9. Ives, B and Olson, M.H. User involvement and MIS success . A review of research. Management Science. 1984; 30, pp586-603.
10. Mumford, E and Henshall D. A participative approach to the design of computer systems design. Associated Business Press, London. 1979.
11. Norman, D and Draper, S .User Centred Design Laurence Esthances. 1986.
12. Polack, F., Whiston, M. and Hitchcock, P. Structured Analysis - draft method for writing Z specifications. Z User Workshop. 1990; Nicholls, J.E. et al. (ed). Oxford, Springer-Verlag.

13. Polack, F and Mander, K Software Quality Assurance using the SAZ Method. Z User Workshop 1994; pp 230-244. Nicholls, J.E. et al. (ed). Oxford, Springer-Verlag.
14. Quang, P.T. and Chartier-Kastler, C. MERISE in Practice. Macmillan. 1991.
15. Rochfeld A., Mehl C., Follot J. Retombees d'une modelisation E-R sur des modeles de traitement MERISE. Genie Logical & System Experts. No 25 Dec 1991; pp 54-60.
16. Semmens, L. and Allen, P. Using Yourdon and Z: an approach to formal specification . Z User Workshop 1990; Nicholls, J.E. et al.. (ed). Oxford, Springer-Verlag.
17. Semmens, L T, France, R B, and Docker, T W G. Integrated Structured Analysis and Formal Specification Techniques Computer Journal. 1990; Vol 35. No 6. pp600-610. Cambridge University Press.
18. Spivey, J M. The Z Notation : A Reference Manual (Second Edition), Prentise Hall International , London, 1992.
19. Swatman, P.A. and Swatman, P.M.C. Managing the Formal Specification of Information Systems. Proc. International Conference on Organisation and Information Systems. Strategies- People-Technologies. Kaltnekar, Z (ed). 1992; pp303-326.
20. Smith, A and Dunckley, L. Human Factors in Software Development. In Human Computer Interaction. Proceedings of INTERACT-95 pp380-383. Chapman and Hall. 1995.
21. Tardieu, A, Rochfield and Coletti, R . Methode Merise: principes et outils; 4eme edition, Editions d'Organisatio, Paris. 1991.
22. Wordsworth. J.B . Software Development with Z. Addison-Wesley. 1992.