# Towards an Interactive Learning Environment for Object-Z

*Selvarajah Mohanarajah, Ray Kemp, and Elizabeth Kemp*
*Institute of Information Sciences and Technology*
*Massey University, New Zealand*

S.Mohanarajah@massey.ac.nz R.Kemp@massey.ac.nz
E.Kemp@massey.ac.nz

## Abstract

We have been engaged in research towards designing a software system called LOZ for learning the object oriented formal specification notation Object-Z. Initially, we conducted a survey to analyse the effectiveness of traditional methodologies for learning Z notation. In LOZ, the semi-formal model UML is used in the intermediate phase between informal textual description and formal Object-Z description. We also employ a refinement unit that produces code from the specification. Based on the cognitive apprenticeship approach, we employ a four-phase instructional model in our system. Persuading the learners to be partially responsible for their own model and allowing them to decide their own levels of control over the learning process are key features of our system.

**Keywords**: Object-Z, Computer Based Learning (CBL), Intelligent Tutoring Systems (ITS).

## Introduction

Reliability is an important requirement for software products. Formal specification is an important vehicle to attain reliability in the software development process. Despite its importance in industry and commerce, formal specification has not been well received by software engineering students. Students often have real problems in understanding this topic. A leading object technology consultant, Martin Fowler, worries that *"formal methods are hard to understand and manipulate, often harder to deal with than programming languages"* (Fowler 1998). One of the main reasons for this difficulty is that students usually get frustrated when attempting to abstract away the initial informal problem description given in some ambiguous natural language text to produce a formal specification. We believe a well designed Computer Based Learning (CBL) system will alleviate this problem and provide great assistance in the learning of formal specification.

There are thousands of CBL systems reported in the literature for a variety of subjects. Within the discipline of software engineering, however, there has been little activity except in the production of systems for teaching computer programming. As a first step towards designing a software system (LOZ) for learning formal specifications, we conducted a survey to evaluate the effectiveness of traditional teaching methodologies and the impact of a formal method tool (Formalizer) in learning formal methods. The results of the survey were encouraging. Our system, LOZ (Learn Ob-

ject-Z), uses a semi-formal graphical language, UML (Booch, Rumbaugh, & Jacobson, 1999), in the intermediate phase between natural language description and formal specification in Object-Z notation (Smith, 2000). To facilitate the students' understanding, a refinement unit to generate program code automatically from the specification will be incorporated in LOZ. Learners build up their own understanding through four phases: conditioning, sketching, building, and exploring. These phases are somewhat modified versions of Allan Collins' original cognitive apprentice model (Collins, Brown, & Newman, 1990). The underpinning learning theory in our instructional model is of a constructivist nature and caters for individualized learning. To address uncertainty, an imprecise probability model is used to characterize the learner. The feedback LOZ gives depends on the complexity of the topic and the learner's level of understanding of the current topic. Learners will be able to view or update their own models.

# Formal Methods

Formal methods are mathematically-based languages, techniques and tools for specifying and verifying systems. The goal is to enable developers to construct software/hardware systems that operate reliably. They may be used to create conceptual models in different phases of the software engineering life cycle: analysis, specification, architectural design, verification, prototyping and testing (Tremblay, 2000). There are more than fifty formal methods available. The Z notation is one of the actively used formal methods (Stepney, Polack, & Toyn, 2003). The notation Object-Z is one of the many approaches for adapting or extending Z to accommodate object oriented features. It is essential to select the right method for the right application. Though formal methods are used at various levels such as software/hardware specification, verification (model checking and theorem proving) and animation, the key notion of formal methods is clearly formal specification. Specification is the foundation on which all further formal work can be done (Tremblay, 2000).

There are more than a hundred tools available for various formal methods. About thirty were reported in the literature for the Z notation only (and there are few reported for Object-Z). Almost all of them provide a syntax-directed editing facility in batch or on-line mode, and include extensive help on language syntax. Symbols may be inserted using icon selection or using tags. The document may be expressed in LaTeX or some other suitable forms. The document may be printed in a pretty format easily. Some of these tools that are considered to be useful for learning formal specification will be discussed in detail later in this paper.

## *Teaching and Learning Formal Methods*

There are many reports in the literature that reveal the various experiences of educators and researchers in teaching formal methods at tertiary level and for practitioners. Jonathan Bowen's experience speaks for the integration of computerized tools in his courses (Bowen, 2000). In their edited book Dean and Hinchey (1996) explain their role-play approach. Gibson and Mery (1998) explain their case-study approach as, *"Rather than concentrating on one particular method, we advocate working on a set of small case studies, using the mathematics in a flexible and intuitive manner, where the students can appreciate the need for formality. Each case study should illustrate, in turn, the need for some fundamental formalism."*

Similarly, we also use a case-study approach for teaching. In this study, LOZ uses appropriate worked examples to illustrate the underlying concepts, notations and techniques. It then gradually allows the learners to solve carefully selected portions of case studies by themselves. The system will help the learners to concentrate on the selected concepts by hiding irrelevant complex processes. The system may provide limited help for learners at various levels until they eventually become expert on the topic.

The following are some of the reasons stated in the literature for students' difficulties in learning formal methods (Mikusiak, Hasaralejeko, & Koronthaly, 1995): (1) insufficient mathematical ability (2) lack of motivation – unable to see the advantages of a complex notation (3) inability to abstract details. To overcome the motivation problem mentioned above, the students can be engaged by demonstrating the tangible benefits of the formalization. For example, the system may automatically produce relevant code for the learner's specification. In LOZ, we use a software animation tool (Refinement Unit) to improve the students' motivation. To address student difficulties with abstraction, we use UML in the intermediate phase. It is assumed potential users of LOZ will have sufficient knowledge and experience in the creation of UML models from informal textual description. Abstraction is an important intelligent tool for creating formal specification from problem description. As Duke and Rose (2000) put it, *"—based on teaching formal methods to students at all levels and to professional software engineers, the central difficulty faced initially is not the comprehension of the mathematical notation, but its application to capture the key abstractions of data and functionality"*.

When students create UML models they are dealing with a level of abstraction (mainly data and some functional abstraction). Therefore, by using a UML model in the intermediate phase we reduce the abstraction burden and allow the users to concentrate on the application of mathematical notation to the key abstractions.

# Related Research

The software systems that are developed for learning formal specifications fall into two categories. The first category covers the formal method tools that are claimed to be useful for learning. These tools are in general developed by researchers in the formal specification discipline. The second category of the systems is developed principally for pedagogical purpose by the researchers in computer based learning discipline. We will discuss the first category CBL systems for Formal Methods in detail now.

## *CBL Systems for Formal Methods*

Despite the number of tools for aiding the development of specifications, we have found no reference in the literature to CBL systems designed for actually *learning* formal methods. MEMO-II (Forcheri & Molfino, 1994) is the only existing CBL system we have found which is related to our research. As declared by its authors, MEMO-II is intended to be used in teaching/learning of programming with University students. It is an education oriented programming environment, which allows users to build programs from formal specifications via interaction with the system. Forcheri et al claim that learning to program requires modelling capabilities. A programming problem may be modelled using two approaches. One is a computational model depending on a programming paradigm, and the other is an abstract model independent of any paradigms. Learning to construct abstract models helps software practitioners to switch effortlessly between different paradigms. MEMO-II follows the second approach, and additionally, Forcheri and Molfino (1994) assert that it also offers facilities to map this abstract model into effective implementations.

From a functional viewpoint, MEMO-II is more than a formal method tool which can provide a syntax sensitive editing/browsing environment, and validating, proving and animating capabilities with its own compiler. From a pedagogical point of view (with regard to learning programming), it is a tool within which learners need to experiment using suitable examples. Learners are left to make their own comparison between specifications and resultant generated code. Furthermore, in order to learn a computer programming language, MEMO-II requires novices to learn a formal specification language, the syntax of which is equally complicated.

Another CBL system that has some linkage with our research is FLUTE (Devedzic *et al.* 2000), although FLUTE is intended to teach formal languages, rather than formal methods *per se*. FLUTE operates in three modes: teaching, examination and consulting. The pedagogical module is responsible for selecting one of these modes depending on the user's choice. There are also student and explanation modules. FLUTE is more concerned about underlying theoretical aspects of formal methods than its software engineering application. It is more useful for computer science students than software engineering students or practitioners. The authors compared FLUTE with other ITSs, but no classroom evaluation is reported.

### *Educational Formal Method Tools*

In the second category, there are various tools for formal specifications claimed to be able to be used for educational purposes (Zbrowser (Mikusiak et al., 1995), ZAL/ZED (Morrey, Siddiqi, Hibberd, & Buckberry, 1993), ZTC/ZANS(Jia 1995b; Jia 1995a) and VisualiZer (Yap 1999). These all require significant initial tutor guidance for novices, and can be used to stress certain topics but none can be regarded as providing a learning system for formal specification. We will discuss only two of these tools in detail.

Zbrowser (Mikusiak et al., 1995) serves as a syntax-directed editor and browser for Z but lacks data type checking, proof or refinement facilities. Mikusiak et al. identify three main difficulties for students learning Z: a requirement for a mathematical background, the complicated notation of Z and the complex structure of Z schema. Zbrowser is designed to overcome these difficulties so that it can be used as a teaching aid. Features like the graphical representation for the Z data types using the table metaphor; the efficient interfacing mechanism to reduce short-term memory loads such as paragraph expansion-contraction facilities; extended subject sensitive error reports; on-line context sensitive help facilities, and problem oriented examples differentiate Zbrowser from many other tools for Z notation. From a pedagogical point of view, all the above mentioned features make ZBrowser an educational tool. Zbrowser was checked in a classroom with 40 subjects and the authors claim that the tests proved that the subjects using Zbrowser performed better than control group in both quality and comprehension of Z specifications.

The package ZAL/ZED is an integration of two tools ZED and ZAL. ZED is a typical formal specification editing tool which supports syntax and type checking and limited semantic checking with some context sensitive help facility. ZAL (Z Animation in LISP) is an animation tool that can generate a prototype in LISP which demonstrates the functionality of the intended system at an early stage of the development process. This package, the author maintains, encourages and facilitates an exploratory (rather than declarative) approach to formal specification, and in turn supports the teaching process as this allows students to use a "try and see" approach. Though no formal evaluation is reported, the authors, based on the students' feedback, provide empirical evaluation and assessment of ZED/ZAL package as follows: "Preliminary results indicate that in addition to the benefits which ZAL provides of increasing students' confidence in their ability to reason in a concrete way about their own specification, it has also provided a test-bed for exploration and experimentation" (Morrey et al., 1993).

## Survey on Z and Formalizer

As noted earlier, there is no software available to teach formal methods. However, there are many software tools used to support teaching formal methods. We initially investigated whether any of these tools could enhance (or hinder) learning formal methods. Formalizer (Flynn, Hoverd, & Brazier, 1989) is such a tool mainly useful for preparing Z specification. It incorporates facilities such as a syntax-sensitive editor and a type/scope checker. A questionnaire was created using a five point Likert type scale to test eight hypotheses about the effectiveness of Z notation and the Formalizer in learning context.

The following 3 research questions were considered critical for the investigation:

- Z is difficult to learn
- Formalizer is difficult to learn.
- Formalizer motivates students to learn Z.

Subjects were third year software engineering students who had completed a basic software engineering course in their second year and had just studied a course on formal methods. Twenty-nine out of thirty two subjects (more than 90%) returned the completed questionnaire. In general, the evaluation of the completed questionnaire reveals that the traditional way of learning formal method in classrooms (despite an experienced and efficient teacher and enthusiastic students) has many drawbacks. Note that the percentages given in the following section refer to the percentages of answers in the bottom or top two points of the scale as relevant.

The main observations are:

- More than 40% report that Z is difficult to learn
- More than 60% report that Formalizer is difficult to learn.
- Only 3% reported Formalizer motivated them to learn Z
- Nearly 70% thought that the help facilities of Formalizer were useless.

Novice users were found to be more exasperated by the interface of Formalizer. More than 90% of the subjects reported that Formalizer is annoying to use. The features such as type-checking that experienced users consider very useful were considered difficult by novices. Only about 25% of the subjects considered some special features in Formalizer were really useful.

None of the subjects disagreed with the fact that the "would be added features" mentioned in the questionnaire (e.g. context sensitive feed-back) for Formalizer would make Z easier to learn. More than 90% of the subjects realized that an on-line tutorial with proper feedback would help them to learn Z notation while using Formalizer (an attachment given with the questionnaire demonstrated the difference between Formalizer's help and the possible feedback that a 'context-sensitive coach' could provide in a similar situation).

We strongly believe that this situation can be improved by using a software application such as LOZ which will provide an effective learning environment and includes remedies for the drawbacks identified.

## Our CBL system - Learn Object-Z (LOZ)

Our system, LOZ, differs significantly from the ones described in the previous sections. It is an environment where the learner builds his or her own knowledge while interacting with the system. The pedagogical approach in LOZ is based on constructivist learning theory and is tailored to individual learner's requirements. Unlike ZED/ZAL, the animation unit in LOZ is not a main component, but, instead, helps to preserve student motivation and interest.

A potential user of LOZ is expected to be a software engineering student or a practitioner who wants to learn this particular discipline. The learner tries to build their knowledge in this discipline by interacting with the system. This may happen in different stages. The system may guide the learner during their investigation. In order to give effective guidance, the system needs to model the learner's current understanding of the subject. In many cases, users' responses are analyzed in a Mentor model to provide appropriate feedback and further actions. For this purpose, the learner model provides the required information about the user and the subject unit provides

domain specific knowledge. The Refinement unit in this design helps to keep the motivation of the learner high. It could automatically generate programming code for some selected formal specification portions when possible.

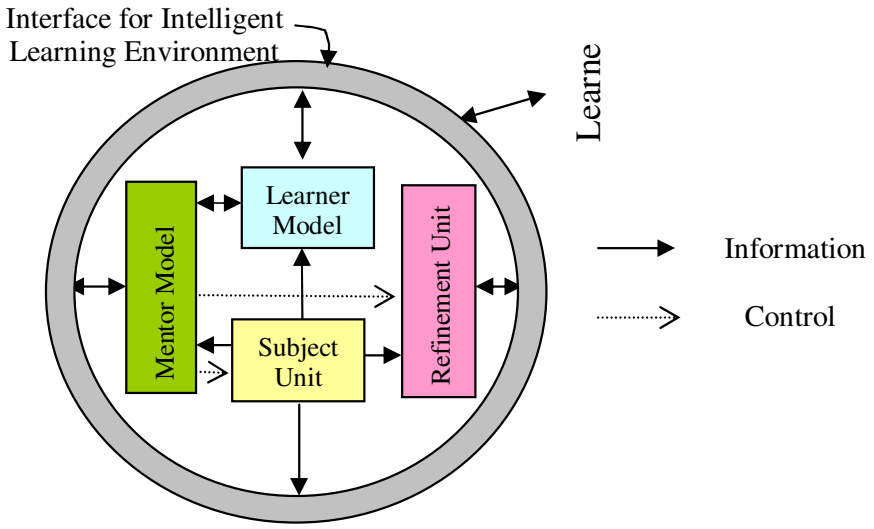


**Figure 1: Architecture of LOZ**

The architecture of the proposed system will be based on the concept of a guided learning environment where the learner has considerable control over their learning process. Figure 1 gives a high level view of its structure.

## *Four-Phase Model*

The basic information in the system is roughly grouped into different lessons. Each lesson consists of several topics and each topic may have several sub-topics. A leaf-topic usually corresponds to various supporting atoms. An atom may be, for example, a video-clip, textual description, portion of worked example or an URL. In the structural view of the system, a preferred order in which these topics could be learned is given as a network. For example, in a lesson "Introducing Formal Specification", a topic is a "Requirement should be specified precisely and unambiguously". Some other sub-topics are given in Figure 2.

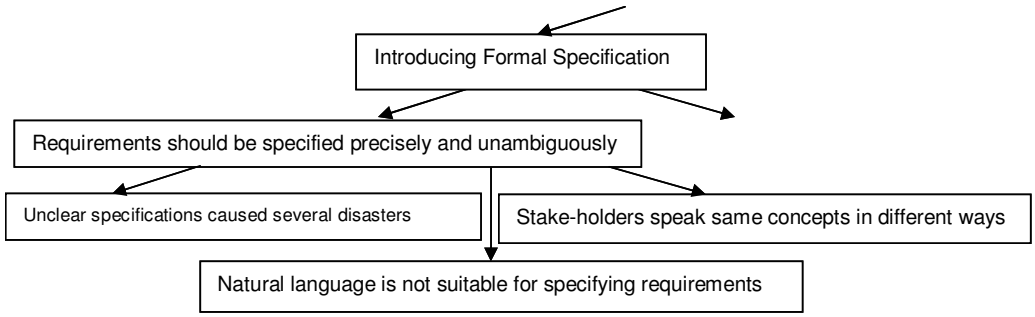


**Figure 2: Curriculum Tree**

We use a four-phase learning model in this research, which is based on the cognitive apprenticeship concept. The original model is based on situated cognition theory that proposes that the

learning is naturally tied to authentic activity, context and culture. This learning model suggests a paradigm of situated modelling, coaching and fading away to empower the learner. The learner is enabled to articulate and reflect on their own knowledge and skills (Collins et al., 1990). Since our system now does not support a collaborative learning process effectively, we simplified the original model to suit our purpose.

In our model, for each leaf-topic the system encourages the learner to go through four phases: conditioning, inspecting, building, and exploring. Initially the learner's cognitive state is conditioned for a new topic. In this stage, the system tries to prepare the learner for the new topic. The learner's previous knowledge of the current topic, if any, should be revised. In the inspection stage, which is similar to Collins et al.'s modelling stage, the learner will form an outline of the topic by examining some worked examples and existing solution models. The user is able to inspect the problems and solutions at various levels. In the building stage, which is similar to the original exploration stage, learners build their own understanding on the topic by solving increasingly challenging problems in related subjects. Finally, in the exploration stage the learner is encouraged and empowered to critically analyse and challenge what they have learned so far and to learn more through other means such as web or email etc. This is mainly through discussions with peers, mentor and other relevant sources. Figure 3 shows a snapshot of the main window in LOZ which consists of main menu, tool bar, navigational bar, case-study access area, status bar, main-learning box and sub-learning box.

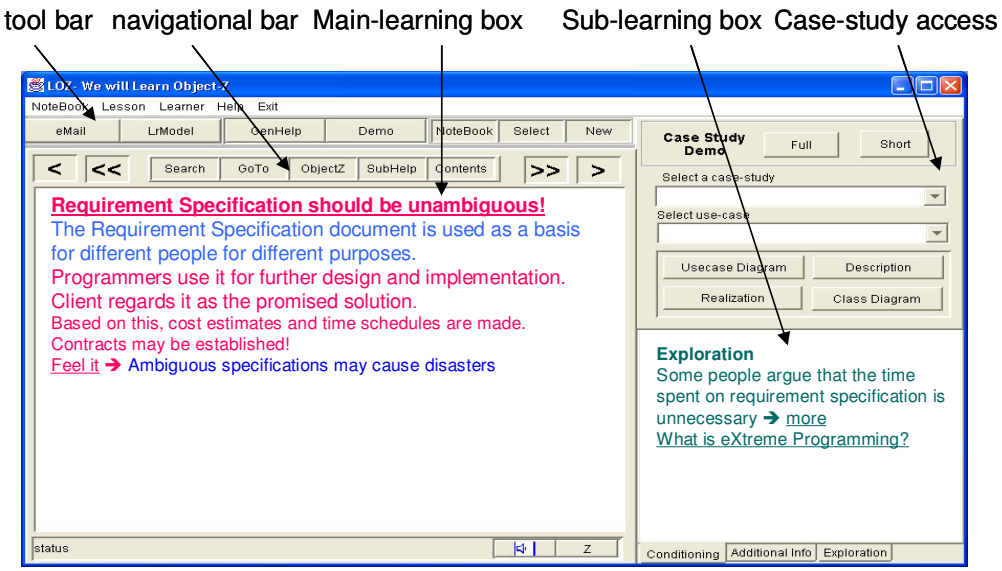


**Figure 3: LOZ – main screen**

Figure 4 shows a class-schema in Z together with the corresponding error report produced by Formalizer (Flynn et al., 1989). Although a professional might find the Formalizer's message useful, the technical level of detail could confuse novices. Consequently, it may be difficult for them to identify the misconception or error from this feedback. LOZ will provide a more novice-oriented response and will guide the student towards an acceptable solution. For this example, it would be pointed out to the learners that the basic type PERSON can only represent a single person and not a collection, and that the set operation, □ (sub-set), can only be used between two sets.

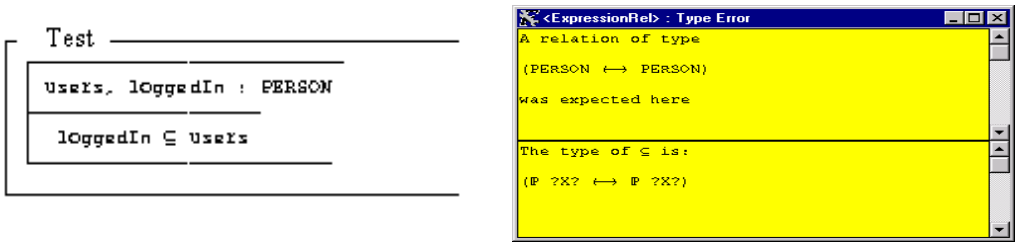


**Figure 4: Student's error and Formalizer's feedback**

## *Learner Model and Feedback*

LOZ is not a passive learning environment. In order to guide learners actively it needs to model them efficiently. The gap between the available evidence and a suitable conclusion will be high in our system. Furthermore, as we allow high-level of learner control we also face the assignment of credit problem. The feedback LOZ gives depends on the learners' level of understanding of the topic, their background knowledge and the complexity level of the topic itself. Table 1 explains various types of feedback (derived from Mason and Bruning (1999)).

**Table 1: Feed back Types**

| Action / Feed back type | Verify | Answer Given | Response Analyzed | Comments |
|---|---|---|---|---|
| **No feed-back** | no | no | no | Total marks only |
| **Knowledge-of-Response** | yes | no | no | Each item is verified |
| **Answer-Until-Correct** | yes | no | no | Repeats same topic |
| **Knowledge-of-Correct Response** | yes | yes | no | For each item answer is given |
| **Topic- Contingent** | yes | yes | no | Directs to the literature |
| **Response-Contingent** | yes | yes | yes | Analyze both answers |
| **Bug-related** | yes | no | yes | Bug identified, directs to self-correction |
| **Attribute-isolation** | yes | yes | no | Focus on Key concepts. |

The decision making process in LOZ for the feedback is explained in Table 2. In order to address the uncertainty efficiently we need to use some imprecise probability model for our learner model. A fuzzy model is suitable for our system as it needs to deal with imprecise user interactions. The fuzzy model was originally suggested for student modeling by Hawkes et al (1990).

**Table 2: Feed back Type and Timing in LOZ**

| | New Learner | | Expert Learner | |
|---|---|---|---|---|
| | **Basic Topic** | **Advance Topic** | **Basic Topic** | **Advance Topic** |
| **Feedback Timing** | Delayed | Delayed | Immediate | Delayed |

| **Feedback Type** | Response Contingent | Topic Contingent | Answer-until-correct | Topic Contingent |
|---|---|---|---|---|
| **Next Topic** | Lower Level | Lower Level | Same level | Lower Level |

The learner model provides an estimate of the current user as a novice, moderate or expert. For a particular case, a moderate type learner may be considered expert or novice temporarily. If a new learner makes a mistake in a basic level topic they will not be given a response immediately, instead the system tries to guide them from a lower level topic until the earlier mistake can be identified. However, response contingent feedback is always possible on a request. This feedback explains why an answer of the user is wrong and argues also for the correct answer. For an experienced learner, if a mistake is made on a basic level topic, LOZ initially assumes it is just a slip and provides immediate feedback. LOZ allows the learner to correct the error immediately. After that, however the same level topic is presented to the learner in order to make an intelligent judgment about their level of understanding.

# Conclusion

LOZ is an intelligent learning system to learn Object-Z. It uses a four-phase learner model that closely matches with the cognitive apprentice model of instruction. It is mainly based on learning by studying examples and solving suitable problems. The system consists of several case studies and their requirement specifications in English as well as in UML. Some case studies are used as worked examples. The system will not be able to generate a formal specification document from a given UML description. If that is a case we only need to learn UML instead of formal methods. The learner will be able to inspect as well as alter his or her own model. Eventually, the learner will reach a phase where they are empowered to challenge their own understanding against various knowledge sources.

The immediate benefit of this research is the design models of a CBL system for a formal specification (Object-Z notation). Secondly, our research will contribute towards identifying efficient instructional design and learner modelling templates or techniques for CBL systems. Moreover, we value the social aspects of the learning activity, and in future, this research will be extended towards designing a collaborative web-based learning environment for formal specification in object oriented notation Object-Z.

Converting a semi-formal model such as UML to a formal model such as Object-Z is a challenging problem. This on-going research will attempt to address several such problems. We hope there will be some Community Object-Z Tool project (similar to the CZT- Community Z Tool project : see (CZT-Project 2003)) initiated in the future to integrate various Object-Z tools, and in that situation we speculate that LOZ will play an important role.

# References

Booch, G., Rumbaugh, J. & Jacobson, I. (1999). *The UML user guide.* Addison-Wesley.

Bowen, J. P. (2000). *Experience teaching Z with Tool and Web support*. London: Center for Applied Formal Methods, South Bank University.

Collins, A., Brown, J., S. & Newman, S., E. (1990). Cognitive apprenticeship: Teaching the crafts of reading, writing and mathematics. In L. B. Resnick, *Knowing, learning and instruction*: *Essays on honor of Robert Glaser* (pp. 453-494). Hillsdale, NJ: Lawrence Erlbaum.

CZT-Project (2003). Retrieved February 17, 2004 from http://czt.sourceforge.net

Dean, N., C. & Hinchey, M., G. (1996). Formal methods and modeling in context. In N. Dean, C. & M. Hinchey, *Teaching and learning formal methods.* London: Academic Press.

Devedzic, V., Debenham, J., & Popvic, D. (2000). Teaching formal languages by an intelligent tutoring system. *Education Technology & Society, 3* (2), 2000.

Duke, R. & Rose, G. (2000). *Formal object-oriented specification using Object-Z*. London: Macmillan.

Flynn, M., Hoverd, T., & Brazier, D. (1989). Formalizer- An interactive support tool for Z. *Fourth Annual Z User Meeting*. Oxford: Springer-Verlag.

Forcheri, P. & Molfino, M. T. (1994). Software tools for the learning of programming: A proposal. *Computers Education, 23* (4), 269-278.

Fowler, M. (1998). *UML distilled; Applying the standard modeling language.* Addison Wesley Longman.

Gibson, P. & Mery, D. (1998). Teaching formal methods: Lessons to learn. IWFM, Ireland.

Hawkes, L. W., Derry, S. J., & Rundensteiner, E. A. (1990). Individualized tutoring using an intelligent fuzzy temporal relational database. *International Journal of Man-Machine Studies, (33),* 409-429.

Jia, X. (1995a). *A tutorial of ZANS -- A Z animation system*. Chicago: DePaul University.

Jia, X. (1995b). *ZTC: A type checker for Z, user's guide.* Chicago: DePaul University.

Mason, J. B. & Bruning, R. (1999). Providing feedback in computer-based instruction: What the research tells us. Retrieved February 17, 2004 from http://dwb.unl.edu/Edit/MB/MasonBruning.html

Mikusiak, L., Hasaralejeko, J. & Koronthaly, D. (1995). Z Browser - Tool for visualization of Z specifications. *ZUM'95 - 9th International Conference of Z Users*. Springer-Verlag.

Morrey, I., Siddiqi, J., I, A, Hibberd, R. & Buckberry, G. (1993). Use of a specification construction and animation tool to teach formal methods. IEEE COMPSAC 93, *The Seventeenth Annual International Computer Software and Applications Conference*, Phoenix, Arizona, USA.

Smith, G. (2000). *The Object-Z specification language.* Kluwer Academic Publishers.

Stepney, S., Polack, F. & Toyn, I. (2003). Patterns to guide practical refactoring: Examples targeting promotion in Z. ZB 2003: Formal specification and development in Z and B. *Third International Conference of B and Z Users*. Turku, Finland.

Tremblay, G. (2000). Formal methods: Mathematics, computer science or software engineering. *IEEE Transactions on Education, 43* (4).

Yap, C. N. (1999). *Visual-Z: A methodology and environment for developing visual formal Z specifications*. Ph.D. thesis.

# Biographies

**Mohan** is an Assistant Lecturer in the Institute of Information Sciences and Technology at Massey University, New Zealand. He has a B.Sc. (Hons) in Mathematics and an M.Sc. in Computer Science. He is currently doing a PhD in Computer Science at Massey University. Mohan has served as a reviewer for several international conferences and been on the program committee for one. His research interests include Computer Based Learning, Artificial Intelligence, Formalizing UML, Object Orientation in Formal Methods and Software Engineering Education

**Ray Kemp** is currently Associate Professor in Computer Science at Massey University, New Zealand. Ray has a B.Sc. (Hons) in Mathematics, an M.Sc. in Computer Science and a PhD. He has published three Books and over fifty papers. Research grants include $1,600,000 as part of team on group research into learning environments (2000) and a Post-Doctoral Position worth $84,000 (1997). Ray has chaired or co-chaired three international conferences and been on the program committee for eight others. His teaching interests include formal methods, declarative programming and artificial intelligence.

**Elizabeth Kemp** is an Associate Professor in the Institute of Information Sciences and Technology at Massey University, Palmerston North. Her qualifications include a Ph.D. in Computer

Science and an MBS with First Class Honors in Information Systems. She was awarded the ANCCAC medal by the Australian Computer Society. Her research interests are wide-ranging, including Computers and Education, Software Engineering Practice, Human-Computer Interaction, Problem Solving and Knowledge Acquisition. Subjects taught over the years include Software Engineering, Database systems, Object-Oriented Analysis and Design, Human-Computer Interaction, Security in Information Systems and Expert Systems.