Designing CBL Systems for Complex Domains using Problem Transformation and Fuzzy Logic

A thesis presented in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science

at Massey University, Palmerston North, New Zealand.

Selvarajah Mohanarajah
2007

Acknowledgement

This research would not have been possible without the support, inspiration, and enthusiasm of my supervisors, Associate Professors R. Kemp and E. Kemp. They gave me the appropriate guidance as well as unbounded freedom, whenever it was necessary. A special thanks to Ray for keeping me as an assistant lecturer for six long years at the computer science division of IIST. I am also grateful to Janina for her support during my final months at Massey. I would also like to thank my former colleagues at IIST including the office staff and technical support staff who has extended their goodwill in numerous ways.

It was Loji, my late wife, whose constant understanding, endless patience, encouragement and love made this mission achievable. While I was studying- during our entire married life- she was awaiting to see my success. It is to her, I dedicate this work. I am also thankful to my daughters Archu and Mathu for their sacrifice and understanding when it was most required. Finally, I am forever indebted to my mother, brothers, Loji's family and our friends, without their unflinching encouragement, this thesis might not have been appeared in its present form.

Publications

The following publications are made out of this research.

1. Mohanarajah, S., R. H. Kemp, and E. Kemp. 'Towards a Computer Based Learning System for Object-Z. Proceedings of the International Conference on Computers in Education 2003 (ICCE'03), Hong Kong.
2. Mohanarajah, S., R. H. Kemp, and E. Kemp. 'Towards an Interactive Learning Environment for Object-Z'. Journal of Issues in Information Sciences and Information Technology Education Vol 1, 2004, pgs 0993-1003 ISSN: 1539-3585, Informing Science, Institute Santa Rosa, California USA. Paper presented at the International Conference on Information Science and IT Education 2004 (InSITE'04), Rockhampton, Australia (awarded Best Paper medal)
3. Mohanarajah, S., Kemp, R.H., Kemp, E., and E.Heinritch. 'Unfold the scaffold & Externalizing Learner Models", World Conference on Educational Multimedia, Hypermedia, and Telecommunications 2005 (ED-MEDIA'05), P. Kommers, G. Richards (Eds.), Association for the Advancement of Computing in Education, Norfolk, USA, pp 4004 - 4009.
4. Mohanarajah, S., R. H. Kemp, and E. Kemp. 'Intelligent Pedagogical Action Selection under Uncertainty'. International Conference on Artificial Intelligence in Education 2005 (AIED'05), Chee-Kit, L, McCalla, G., Bredewreg, B. and J. Breuker (Eds), IOS Press, Amsterdam, Netherlands, pp 881-883
5. Mohanarajah, S., Kemp, R.H., and Kemp, E.. 'Opening a Fuzzy Learner Model', Proceedings of the Workshop (LEMORE) at the International Conference on Artificial Intelligence in Education 2005 (AIED'05), Amsterdam, Netherlands.
6. Mohanarajah, S., R. Kemp, H., et al. (2006). Adaptable Scaffolding - A Fuzzy Approach. LNCS 4053 - Intelligent Tutoring Systems (ITS' 6). M. Ikeda, K. Ashley and T. W. Chan. Berlin, Springer-Verlag: 604-614.

dedicated to
LOJI

Abstract

Some disciplines are inherently complex and challenging to learn. This research attempts to design an instructional strategy for CBL systems to simplify learning certain complex domains. Firstly, problem transformation, a constructionist instructional technique, is used to promote active learning by encouraging students to construct more complex artefacts based on less complex ones. Scaffolding is used at the initial learning stages to alleviate the difficulty associated with complex transformation processes. The proposed instructional strategy brings various techniques together to enhance the learning experience. A functional prototype is implemented with Object-Z as the exemplar subject. Both objective and subjective evaluations using the prototype indicate that the proposed CBL system has a statistically significant impact on learning a complex domain.

CBL systems include Learner models to provide adaptable support tailored to individual learners. Bayesian theory is used in general to manage uncertainty in Learner models. In this research, a fuzzy logic based locally intelligent Learner model is utilized. The fuzzy model is simple to design and implement, and easy to understand and explain, as well as efficient. Bayesian theory is used to complement the fuzzy model. Evaluation shows that the accuracy of the proposed Learner model is statistically significant. Further, opening Learner model reduces uncertainty, and the fuzzy rules are simple and resemble human reasoning processes. Therefore, it is argued that opening a fuzzy Learner model is both easy and effective.

Scaffolding requires formative assessments. In this research, a confidence based multiple test marking scheme is proposed as traditional schemes are not suitable for measuring partial knowledge. Subjective evaluation confirms that the proposed schema is effective. Finally, a step-by-step methodology to transform simple UML class diagrams to Object-Z schemas is designed in order to implement problem transformation. This methodology could be extended to implement a semi-automated translation system for UML to Object Models.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

With recent advances in information and communication technologies, the educational needs of society demand radical changes in learning practices. The traditional notion of primary-secondary-tertiary education followed by professional work has become inadequate. The current knowledge-driven, competitive, marketing economy requires that knowledge and training be available on-demand, offered in a flexible mode and be life-long. Computer Based Learning (CBL) systems are expected to play a major role in catering for these changes (Broberg 1999).

The research related to CBL systems has more than a four decade long history. There have been thousands of systems reported in the literature related to a variety of disciplines. Learning theories play important roles in determining their fundamental architectures. Current CBL systems often incorporate the notion of 'active learning', which has been promoted by researchers from different disciplines for many years under different names (Laurillard 2006), for example, Constructivism (Piaget 1968). Although, there are different theoretical positions attributed to Constructivism, all of them basically consider learning to be an active process and knowledge is constructed by the learner, based on past experience (Kanuka *et al.* 1999).

Constructionism, an important instructional theory founded on the constructivist learning theory, has been used as a basis for several CBL systems (Papert 1991). For efficient knowledge acquisition, this theory argues that learners should be engaged in personal meaningful creative activities. Learning by doing is better than learning by receiving. Similar to games, there will be an aim towards a goal, for example, in LOGO, the learners are encouraged to write simple computer programs (Papert 1979). Problem transformation is one of the constructionist approaches that matches with both basic ideologies of constructivism, where the learners are encouraged to convert between representations

(Kemp *et al.* 2001). In Leopard Tutor (Kemp *et al.* 2003), for example, the learners are encouraged to create class diagrams from given computer programs. The transformation process, the authors argue, will help the students to learn OO programming by bridging the gap between program understanding and program construction.

Only a few CBL systems are being used in real educational settings. One of the main criticisms of interactive environments is that they are not effective for less able learners (McArthur *et al.* 1999). Meanwhile, research in Learner models (Greer *et al.* 1994) attempts to provide suitable support, tailored to individual learners. Even some interactive environments, for example, SMITHTOWN (Shute *et al.* 1990b) embed simple Learner models and basic expert systems. Nevertheless, designing efficient Learner models is considered challenging in CBL circles. Self (1990) discusses some guidelines for designing optimal Learner models including 'externalising'. Recently, after nearly two decades, the idea of opening Learner models is gaining acceptance (Kay *et al.* 2005).

It is difficult enough for a human teacher to gauge the standard of a student, but a computer, with its restricted forms of communication has even more problems and therefore, uncertainty is inevitable in Learner models. Handling uncertainty is considered challenging in the CBL community. There have been several numerical techniques employed for uncertainty management in Learner modelling (Jameson 1996). Most of these techniques fit into either one of the two major paradigms: Bayesian network-probability theory (Pearl 1988) or fuzzy logic- possibility theory (Zadeh 1994)).

1.2 Research Questions

Inherently, some disciplines are complex to learn. Building CBL systems that facilitate 'active learning' for complex disciplines is considered challenging. The complexity, however, could be reduced by employing educational aids such as scaffolding (Hogan *et al.* 1997). In order to teach a complex discipline, when a similar but relatively easy domain is already known to the learner, scaffolding may be applied using a gradual transformation of learning materials from less complex to more complex disciplines. This research investigates the ways and means of designing CBL systems for certain complex domains that support active learning based on problem transformation and employing scaffolding techniques in order to reduce the difficulty associated with learning such

disciplines. As an exemplar, an object oriented formal notation Object-Z (Smith 2000) and the semi-formal notation UML (Booch *et al.* 1999) are selected as the subject and support domains, respectively. The transformation process from UML models to Object-Z models will be used in order to design suitable artefacts for active learning.

Learning complex domains, even with scaffolding, will not be easy unless some kind of assistance is provided and tailored to individual learners. To the knowledge of the author, there has been nothing reported in the CBL systems research literature about incorporating adaptability in the systems that are based on problem transformation. Furthermore, for uncertainty management, most of the existing approaches use rigorous mathematical formulas based on probability theory (Jameson 1996). Possibility theory, though it is more natural for human understanding, has not been used to its full potential for this purpose. Both probability and possibility theories have their own advantages and disadvantages (Hopgood 2000).The primary question in this research is, whether it is possible to devise a simple but efficient Learner modelling strategy that is based on both possibility and probability theories (which complement each other), in order to make the scaffolding process adaptable even under uncertain situations.

Furthermore, opening learner models that use complex mathematical concepts such as Bayesian Networks for uncertainty handling is considered challenging (Zapata-Rivera *et al.* 2004). This research examines whether this is also true for fuzzy Learner models. In fuzzy models, the variables resemble the underlying real world entities; and therefore, the explanation does not need complex additional visualization systems. This research will also investigate how a fuzzy Learner model can be opened easily and efficiently.

As mentioned previously, Object-Z and UML are selected as exemplar domains. Formal specification is an important vehicle to attain reliability in the software development process. This is often taught in the third year at universities (ACM-SE 2004)). An efficient CBL system for Formal Methods would be useful for both universities and industry. A software practitioner may use it as a job-training companion. Despite its importance in industry and commerce, formal specification has not been well received by software engineering students. Martin Fowler, a leading object technology consultant, states, *"Formal methods are hard to understand and manipulate, often harder to deal with than programming languages"* (Fowler 1998, p. 12).

One of the main reasons for the difficulty is that students usually get frustrated since they need to abstract away the initial informal problem description, given in some ambiguous natural language text, to produce a formal specification, whilst representing those abstractions in some complex mathematical notations. Fortunately, the semi-formal notation UML (Booch *et al.* 1999), which is usually taught in the second year at universities, can provide support. Learning a formal notation based on a known semi-formal notation could be a definite advantage. The basic question now becomes narrower: whether the above fact could be exploited, in order to design an instructional technique based on problem transformation and the scaffolding process for a CBL system, which would alleviate the difficulty in learning Object-Z.

To facilitate scaffolding, a complete step-by-step methodology is required to transform UML models to Object-Z models. There are many attempts reported in the formal method literature to formalise UML, using Object-Z or other formal methods (Amalio *et al.* 2003). However, there is no concrete method available to transform UML models to Object-Z models. This research attempts to formulate such a transformation methodology, which can be used for an interactive, semi-automated conversion of specifications from UML to Object-Z.

Finally, scaffolding usually depends on formative assessment. For formative assessment, Constructed Response tests are preferable to Multiple Choice (MC) tests (Simkin *et al.* 2005). Comparatively, Constructed Response tests can measure the knowledge state accurately. However, in CBL systems, conducting Constructed Response tests for certain subjects may be difficult, due to the nature of the domain (e.g. complex notations). Alternatively, MC tests may be preferred in such a situation. However, traditional MC tests cannot measure partial knowledge. The existing confidence based MC test schemes (Gardner-Medwin 2005), measure partial knowledge, but they are not suitable to provide item-specific feedback. This research also attempts to devise a confidence-based MC test schema that is suitable for item-specific feedback in formative assessments.

1.3 Objectives

The key objectives of this research are:

- To analyse the factors associated with designing an efficient CBL system for certain complex disciplines, together with appropriate features to support the active learning process, based on problem transformation and scaffolding.

- To propose an instructional strategy that includes suitable design artefacts to facilitate active learning and boost the motivation of the learners, by alleviating the difficulties associated with learning such complex disciplines.

- To devise a simple, but efficient, Learner modelling strategy for pedagogical action selection[1] under uncertainty based on artificial intelligence techniques (by combining probability and possibility theories).

- To investigate the pros and cons, and the ways and means of opening such a Learner model easily and efficiently.

- To devise a step-by-step methodology to transform UML notation to Object-Z notation. This methodology should be general enough to be used in any semi-automated translation system, from a UML model to an Object-Z model.

- To devise a confidence-based MC test scheme, in order to provide item-specific feedback in formative assessments.

- To design and implement a prototype, in order to test the proposals presented in this research. This will be constructed in evolutionary fashion with reusable

[1] The term Pedagogical Action Selection, first coined by Mayo *et al* (2001), describes the acts of both selecting the remedy (usually feedback after a misconception is identified) and selecting the next pedagogical action (usually curriculum sequencing, involving one or more actions: revising, moving forward, testing etc).

components, so that a fully functional CBL system may be implemented by simply extending and modifying this prototype.

- To evaluate the prototype and then interpret the results in order to test the effectiveness of the research proposals.

1.4 Scope

Anderson et al (2000) identify four dimensions of knowledge: factual, conceptual, procedural, and meta-cognitive. The first three dimensions are specific to particular learning domains, and the fourth, meta-cognitive dimension, is specific to a particular learner and refers his/her awareness of their own knowledge and cognitive processes. Factual knowledge includes definitions, syntax, and terminologies pertinent to the underlying discipline, conceptual knowledge includes theories, classifications and semantics, and procedural knowledge includes algorithms, methodologies and techniques (*ibid*). Based on this classification, a typical Object-Z course may be divided into three parts. The first two parts may cover the factual and conceptual knowledge dimensions. The first part may include the fundamentals of class schema (such as syntax and semantics), and the second part may cover the object oriented features specific to class schema. The final part may cover the procedural knowledge. It may include case studies and related explanations for specifying real problems using class schemas. A learning system for Object-Z could be designed in two phases: the first two parts (includes factual and conceptual knowledge) can be covered in the first phase, and the third part which includes the procedural aspects may be created in the second phase. This research covers the first phase extensively.

1.5 Methodology

Initially, in order to acquire background knowledge pertinent to refine the problem and to identify potential solutions, relevant literatures need to be reviewed. After that, as a first step towards designing a software system for learning formal specifications, a survey will be conducted to evaluate the effectiveness of traditional teaching methodologies and the impact of an important formal method tool in learning formal methods.

The next step will involve conceptualizing appropriate strategies (e.g. an instructional strategy) in order to construct the components of the learning system that facilitate active learning whilst reducing the complexity of the subject matter in general. The most important task will be formulating an efficient Learner modelling strategy under uncertainty. Particular features of the selected domains (here they are Object-Z and UML) will be exploited to incorporate problem transformation during the scaffolding process. In the meantime, a suitable MC test schema will be designed with the intention of providing rich feedback. Unlike the traditional MC test scheme, the proposed scheme should be able to measure confidence levels on each answer options

Following this design, a functional prototype will be developed to ensure that the overall design is actually workable. This research could be extended in several directions. Therefore, the prototype will be constructed in an incremental fashion: the key design artefacts will be created as separate reusable components. Though the tasks mentioned above may reflect a waterfall model, they will be actually completed in an incremental iterative style. For example, literature will be reviewed, though at a modest level, even in the prototyping phase.

Finally, an evaluation of the prototype will be conducted to test the key concepts developed in this research. A group of students will be asked to participate in the evaluation process. For this purpose, the system will be amended to maintain certain log files. A questionnaire will also be administered.

For the second phase, a full design is out of the scope of this research. However, a step-by-step algorithm to completely transform a UML model to an Object-Z model will be developed. A suitable case study will be used to illustrate this algorithm.

1.5 Thesis Outline

Following this introductory chapter, the next three chapters comprise the literature reviews. Chapter 2 outlines the important learning and motivational theories, and provides general information on CBL systems. It also discusses the variety of Learner models and their mechanisms for uncertainty handling. The role of confidence-based MC tests in

formative assessment is also discussed. Chapter 3 outlines various CBL systems for computer programming and formal methods. A discussion of formal method tools, that claim to be facilitating learning, is also included for completeness. Chapter 4 includes an outline on the CBL systems that use problem transformation techniques. It further outlines the key features of the UML and Object-Z models, which are used as the exemplar domains. Formalising semi-formal models is an important research discipline in its own right. Finally, a discussion on transforming UML to Object-Z models is also included in Chapter 4.

Chapters 5 and 6 present the nucleus of this research. The fundamental architecture of the proposed CBL system (Phase I) is discussed in Chapter 5. The key functional units, their conceptual descriptions, and their integration are discussed in detail. Chapter 6 is totally devoted to the proposed Learner model design. Chapter 7 discusses the prototype development. The evaluation process, results and discussions are provided in Chapter 8. Chapter 9 explains the key issues related to designing Phase II of the CBL system. A step-by-step methodology is proposed in order to transform simple UML class diagrams to Object-Z schemas. A case-study is used to illustrate the strategy. Chapter 10 includes the conclusion, benefits, and potential extensions of this research. A list of publications arising out of this research is also included. All the supporting material is included in appendices: the code of the prototype, the materials related to the evaluation process (questionnaires, test questions, results etc.), and some screen shots of the prototype.

Chapter 2

Computer Based Learning Systems

2.1 Introduction

In the last three decades, several kinds of computer-based learning systems have been reported in the literature. The following list includes some of the common types found (Shute *et al.* 1995): Computer Aided Instruction (CAI), Intelligent-CAI (ICAI), Intelligent Tutoring Systems (ITS) or AI-ED systems, Micro Worlds, Interactive Learning Environment (ILE), Bi-Modus Learning Environment or Intelligent Learning Environment, Computer Supported Collaborative Learning, and finally e-Learning Systems.

Unless otherwise specified, the term *Computer Based Learning* (CBL) system will be used in this dissertation to refer to all of the above mentioned kinds of learning systems. The term *e-Learning system* although more typical, is avoided here as it is more suited to web-based systems and also includes more general tools such as Learning Management Systems (e.g. Black Board (2000)).

In Section 2.2 important learning theories are outlined. Section 2.3 gives an overview on various kinds of CBL systems. The next section discusses the impact of different learning theories on CBL systems. Motivation is essential for learning. Section 2.5 describes the links between motivation and learning. The Cognitive Apprenticeship Model, one of the important instructional strategies, is discussed in Section 2.6. Section 2.7 outlines the role of authoring systems in CBL systems design. The notion of pedagogical action selection is outlined in Section 2.8. The Learner models and the key uncertainty handling techniques are discussed in Section 2.9. Finally, Section 2.10 describes the role of multiple choice questions as a formative assessment tool in CBL systems.

2.2 Learning Theories – An Overview

Philosophies of knowledge entail philosophies of learning. Efficient teaching strategies and instructional designs may be derived from learning philosophies. Whether it is computer based or not, instructional design should be based on some learning principles. During different eras several learning philosophies have been proposed for knowledge and its acquisition. All of those philosophies are grouped primarily under the following schools of thought: Behaviourism, Cognitivism (Information Processing), and Constructivism.

2.2.1 Behaviourism

The Behaviourist theory (Pavlov, 1849-1936 to Skinner, 1904-1990) centred solely on observable behaviour changes and did not take into account anything occurring in one's mind. The mind was viewed as a "black box" (Mergel 1998). For learning to occur, this theory demands that a new behavioural pattern should be repeated until it becomes automatic. Operant conditioning (Skinner 1954) is one of the notable examples of learning theories of this kind.

2.2.2 Cognitivism

Cognitive theory concentrates on the thought process behind behaviour. Behavioural changes, if any, are considered as gauges of changes in the mind (due to learning). Cognitive theory is built on the following key concepts: an internal knowledge structure (schema), the information processing model, and the rules that govern learning effects (e.g. meaningful information is easy to learn) (Reed 2003). In the context of learning, Cognitive Load Theory (Sweller 1994) is one of the theories associated with this paradigm.

The schema concept, first proposed by Bartlett (1932), has been used for decades – and is still being used - in AI and Cognitive research to represent knowledge structures that include several related pieces of information in an encapsulated form.

The information processing model of human memory distinguishes between the following three distinct memory modes: sensory registers, short-term memory (STM- primary memory), and long-term memory (LTM- secondary memory). Figure 2.1 gives the stages of an information processing model. The arrows indicate the possible directions of information flow.

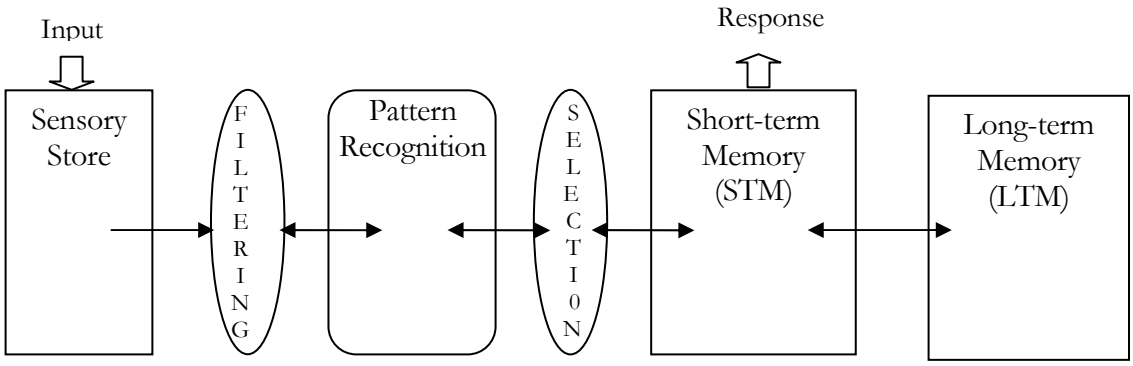


Figure 2.1 Stages of an Information Processing Model (after (Reed, 2003, p. 71))

The division of the stages above are influenced by the computer metaphor. The information received from our senses is briefly stored in sensory store (separate registers for different senses, including visual, auditory, and haptic). Attention plays an important role in filtering out unimportant information. The filtered information flows through to the pattern recognition stage. Familiar patterns are recognised. An unrecognised pattern may be stored if important. Attention may play a further role in selecting appropriate information to pass to the STM. While learning, and in many other tasks, the information can flow back from LTM to STM in order to process the current and stored information simultaneously. Therefore, the STM is also called working memory. However, it is severely limited by storage capacity and retention duration. Miller (1956) shows that, on average, people can keep only a limited number of items (seven plus or minus two) at a time in the short-term memory. This feature limits people's capacity severely. He also shows that chunking (organising similar items in groups) is a useful mental tool to overcome this limitation.

The other limitation, short-term retention, is caused by interferences (or decay). Reed, while discussing proactive and retroactive interferences in STM, put it, *"….whenever possible, we should try to reduce interference by ordering material in an appropriate sequence. Items that are likely to interfere with each other should be studied at different*

times rather than during a single session" (Reed, 2003, p. 75). The LTM has a large capacity and holds information for a longer time; however, storing (learning or transforming information from STM to LTM) requires considerable effort. Atkinson *et al*. (1968) proposed some control processes (rehearsal, coding and imaging) to improve knowledge acquisition and retention.

Cognitive Load Theory (Sweller 1994) distinguishes between two types of cognitive loads: intrinsic and extraneous. The *intrinsic* cognitive load depends not only on the number of concepts to be learned at once but also the extent to which it demands the simultaneous interactions of constituent elements. The *extraneous* cognitive load, on the other hand, depends on how the learning material is presented (Sweller 1988).

2.2.3 Constructivism

Constructivism claims that one's knowledge is constructed by oneself from past knowledge; that is, knowledge cannot be merely transferred, and learning is an active (not a passive) process. There are different epistemological positions in constructivism. Kanuka *et al*. (1999) identify four subdivisions in the major forms of Constructivism along two dimensions. The first dimension is based on whether the theory argues for single reality (objective) or multiple realities (subjective). The second dimension is based on the question of whether the knowledge is constructed individually or socially.

The cognitive constructivists believe that there is only one true external reality and knowledge is individually constructed (Kelly 1955; Glaser 1990). The social constructivists also believe in the single reality (but realise there are multiple perspectives and then construct a single reality by shared meaning through social negotiation), but argue that knowledge is constructed in social interaction (Vygotsky 1978).

Salviati puts it,

> *"As I understand it, this view holds that learning is infinite and not subject to the sorts of analysis favoured by objectivists except in the most trivial cases. …The role of education in a constructivist view is to show students how to construct knowledge, to promote collaboration with others to show the multiple perspectives that can be brought on a particular problem, and to arrive at self-chosen positions*

> *to which they can commit themselves, while realizing the basis of other views with which they may disagree"* (cited in (Cunningham 1992, p. 36)).

The radical constructivists argue that there are multiple realities but believe that knowledge is constructed individually, independent of context (Jonassen 1990). Finally, the situated constructivist believe that there are multiple realities and argue that knowledge is constructed as a social process (Spiro *et al.* 1992). This stand is generally known as *Situated Cognition* (Brown *et al.* 1989; Lave *et al.* 1990). Those who adopt this position emphasise that knowledge does not exist in one's memory but rather is maintained in the external, social world and emerges from interaction with the environment. They argue that knowledge cannot be abstracted and will not be transferable; and therefore, one should learn processes rather than products (Brown *et al.* 1989; Lave *et al.* 1990).

In general, the following five characteristics of long-term knowledge acquisition are abstracted in Hatano (1998); knowledge is acquired by construction (not by transaction), knowledge acquisition involves reconstruction (not gradual increment), knowledge acquisition is constrained both internally (one's previous knowledge) and externally (such as shared language), knowledge is domain specific (in problem solving, only relevant knowledge is accessed) but it can be transferred or generalised and knowledge acquisition is "situated" (it consists not only of abstract rules and laws but also of personal experiences).

2.2.4 Beyond Situated Cognition- Academic Knowledge

Laurillard (2002) argues that academic knowledge, unlike every day knowledge, has specific features. As she puts it *"Academic knowledge is located in our experience of our experience of the world"* (Laurillard 2002, p. 21). Therefore, she argues, *"Situated cognition is attractive in well-chosen situations, but one of the reasons that education evolved the way it has over the centuries is that situated cognition is not enough"* (Laurillard 2002, p. 16). She argues that, for academic knowledge, acquisition abstraction and formalisation are inevitable. It can be noted that the subject domain of the proposed CBL system in this research is highly formal.

2.3 CBL Systems – An Overview

Using human-made devices for learning is not a new idea. It has its roots in the 1920s when Pressey (1926) used his mechanical teaching machine for MCQ tests. This device was based on the theory of connectionism (Thorndike 1913), which represents the stimulus-response model of behavioural psychology. Three decades later, in the 1950s, Skinner used the idea of teaching machines, which modelled his operant conditioning theory (Skinner 1954), as a basis to create an instructional methodology known as *programmed instruction* (Skinner 1958).

2.3.1 From CAI to ICAI

With the introduction of computers, the early computer-based learning systems that originated from programmed instruction were called *linear programs*. These early linear programs were monotonous and boring until (Crowder 1959) introduced multiple paths in programmed instruction to alleviate their rigidity. The next leap in CBL evolution was the development of *generative systems*, which could generate appropriate problems tailored to the ability of the learner (Uhr 1969). In general, the domain knowledge in these initial systems was not separable from the instruction code. Numerous systems were developed on this basis until Carbonell (1970) separated knowledge from control code. Inspired by contemporary AI research (Quillian 1968), he used the *semantic net* for knowledge representation instead of pages of texts. Carbonell also included a Learner model in his system. The early systems are called CAI, and the new systems are have an added 'I', in the abbreviation - ICAI (Carbonell 1970).

2.3.2 From ICAI to ITS

Triggered by the trend pioneered by Carbonell, several other researchers started to investigate the possibilities of incorporating some other AI techniques into CBL systems (e.g. planning, plan-recognition, natural language processing, etc.). Principally, all of these attempts had the aim of making the systems intelligent; that is, able to predict the status of the learner in order to select the most suitable pedagogical action, and to mutually interact with the learner in a user-friendly manner. The resultant systems are called ITS or AI-ED systems.

A generic ITS system consists of four key components; domain model, Learner model, teacher model and interface unit[2] (Hartley *et al.* 1973). Figure 2.2 describes the functionality of a typical ITS system (Shute *et. al.* 1995). The domain model includes subject knowledge. It is not just a sequence of text pages- it may contain declarative as well as procedural knowledge (as facts and rules), or it may include a rich semantic net. Additionally, domain models may include reasoning mechanism.

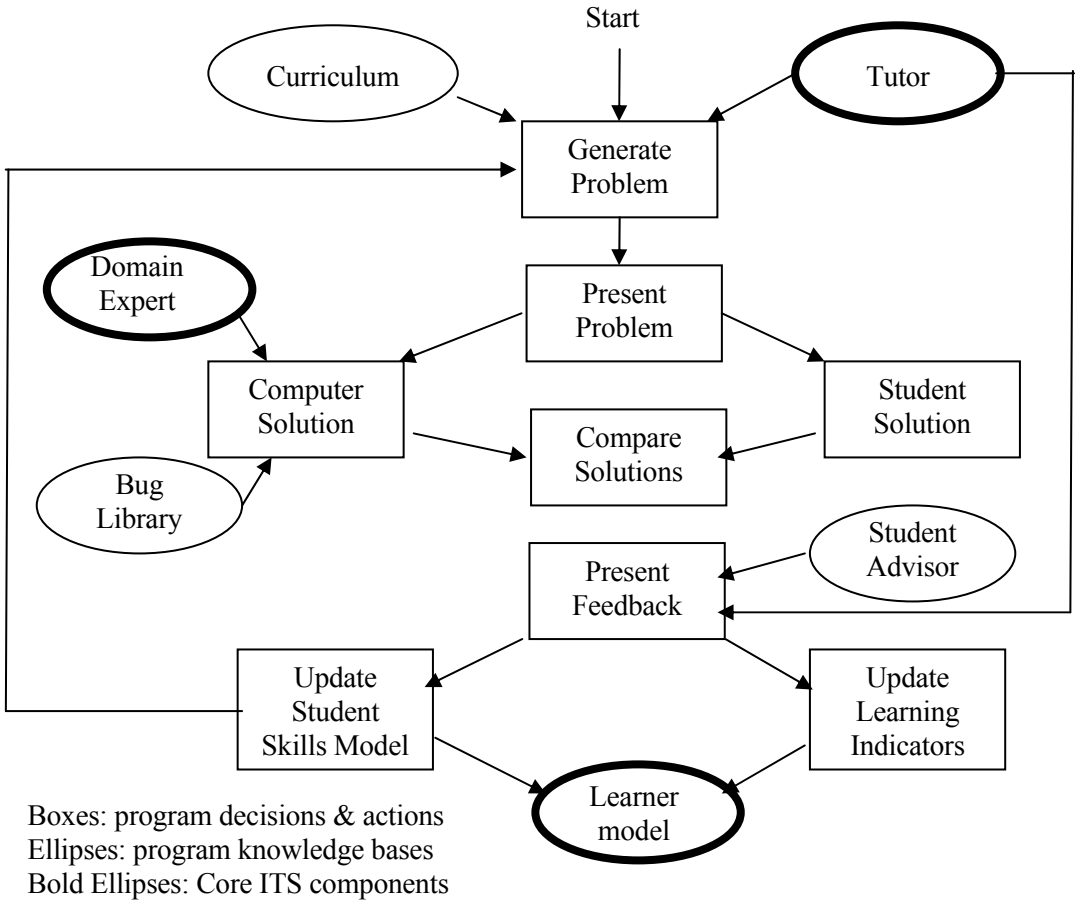


Figure 2.2 Intelligent Tutoring Systems (after (Shute *et al* 1995))

The Learner model is the most debated functional unit in ITS literature. Ideally, it should be able to assess the knowledge level of students during their learning process. However, the accuracy of this estimate is limited. Based on this estimation, the teacher model decides the level of pedagogical actions such as feedback (if any) and curriculum

[2] An ideal ITS needs to have at least four intelligent components; subject expert, teaching expert, accurate Learner model and an efficient natural language processing interface. If we have a good subject expert system the subject can be totally automated; and therefore, we do not need to learn this subject. This situation is referred to as 'catch-22' (McArthur *et al.* 1999)

sequence. The teacher model simulates the decision making process of a typical teacher (Shute *et. al.* 1995). The interface unit may be a rich visual interactive environment. Obviously, natural language based interfaces could closely resemble a human teacher. More discussions on Learner model and pedagogical action selection process are included in Sections 2.8 and 2.9 respectively. Section 2.10 discuss some issues related to using a particular formative assessment format (Multiple Question Test) in ITS.

2.3.3 From ITS to ILE

Many AI enthusiasts initially thought that creating intelligent tutors to replace human tutors would be an easy task. As researchers gradually realized this was not the case, they concentrated on the practical application of AI techniques to learning systems. Self noted this situation as

> *"A great deal of opprobrium was being heaped upon ITS research from all directions including from within itself, for many leading figures (such as Brown, Wenger, Clancey, Sleeman and Soloway) considered that AIED research was misguided, relying as it appeared to do on out-moded philosophies of knowledge and learning"* (Self 1999, p. 354).

In the midst of many such critiques, another architecture called ILE emerged (McArthur *et al.* 1999). Basically, the ILE is used to denote all the types of CBL systems that incorporate *active learning* features and allow learners to explore their environment freely. This type of system may not use AI totally; and therefore, it may not employ a Learner model and explicit knowledge representation.

Discovery World

Some researchers believed that computer programming might be used as a basis for learning all the plan-based studies (Papert 1972). However, the attempts to use computer programming in this manner failed (Palumbo 1990). Nevertheless, this thinking paved the way for a special kind of ILE called the *micro-world* or *discovery world*. This type of system, in general, offers a limited simulation environment within which the learner may engage in constructing some artefacts or perform certain experiments by changing parameters, and this, in turn, enforces active learning (Papert 1972). Computers are now

capable of providing elegant, efficient, aesthetic and interactive environments. A carefully designed environment may be capable of engaging even an unenthusiastic student in active learning activities. Nevertheless, many critics argue that this type of learning system is suitable for only the kind of learners who like to learn independently (Shute *et al.* 1995).

2.3.4 From ILE to Bi-modus Learning Environment

Critics of ITSs complain that they are less learner-controlled, and the Learner model is not usually robust and efficient, whereas the critics of ILEs complain about lack of learner support (Shute *et al* 1995). A realistic approach that lies on the midway between the above two positions comprises limited intelligent Learner modelling using AI techniques to guide learners while they explore or experiment with the environment. These systems provide facilities for an active learning environment with unobtrusive guidance to the learner. Many names have been used for slightly different systems that are based on this ideology: the list includes, Guided-Discovery Environment (Elsom-Cook 1990), Mixed-Initiative Systems (McArthur *et al.* 1999), Bi-modus Learning Environment (Otsuki 1993) and Intelligent Learning Environment (Shute *et al.* 1990a).

2.3.5 Collaborative Learning Systems

Another important type of CBL system that incorporates social learning features is called a *Computer Supported Collaborative Learning* system (Koschmann 1996). Human beings are social animals. We learn many things from the environment. The philosophy behind Collaborative Learning systems is that learning is a social process (Lipponen *et al.* 2004). The so-called *solo CBL* systems are designed for individual learners. Thinking aloud is the basis for social interaction. Computers and networks are now powerful enough to provide rich collaborative learning environments. At the lower level, email may be used as a collaborative tool. The capacity of the Internet, web, LAN and other platforms for collaborative learning is being investigated constantly. Co-operative learning and collaborative learning are different. In co-operative learning, each member is responsible for a pre-determined portion, whereas in collaboration, all the members work together on a shared commitment in a coordinated manner. Lehtinen *et al*. (1999), in their influential review, tabulate the effects of different types of Collaborative Learning systems.

2.3.6 E- Learning

The term *e-Learning system* is being used to refer to a wide range of learning systems including web-based systems. Laurillard defines it as *"the use of any of the new technologies or applications in the service of learning or learner support"*(Laurillard 2006, p.72). She is enthusiastic about the potential of the learning environments (all e-Learning practices, in general) for active learning. She states *"E-Learning could be a highly disruptive technology for education – if we allow it to be. We should do, because it serves the very paradigm shift that educators have been arguing for throughout the last century"* (*ibid,* p. 73). She also worries that *"despite the fabulous advance of the technology, there are surprisingly few real-time interactive simulation games in education"* (*ibid,* p. 84). Obviously, without some sort of centralised initiatives, it is impossible for individual institutions to create a highly sophisticated learning environment for a particular course.

2.4 Learning Theories and CBL Systems

The early systems such as PLATO (Bitzer *et al.* 1961) and TICCIT (Faust 1974) were based on behaviourist theory. In particular, Merrill's instructional theory (later called Component Display Theory (Merrill 1980)) provided the basis for the lesson design in TICCIT. A particular learning task was decomposed through analysis into specific measurable tasks and tests were designed to cover each subtask in order to guide the teaching process in drill-and-practice fashion. The theory avoids activities of mind. There is no explicit knowledge in the system. And the knowledge state or misconceptions of the students are not considered. Most of these systems were developed by educational theoreticians and many of them are still being used in real teaching tasks.

Later, with the introduction of AI to CBL systems, a number of ITSs were developed based on cognitive theory. For example, a variety of ITSs were developed based on ACT* theory (Anderson 1993) and used "model tracing" for student behaviour modelling (e.g. LISP Tutor (Anderson *et al.* 1985)). Expert systems are also used as a basis in some CBL systems (e.g. GUIDON-(Clancey 1986)). Another approach based on cognitive theory is the "buggy model" (impasse or failure-driven) (e.g. PROUST (Johnson *et al.* 1985)). A notable variation on this model is the "mal-rule approach" (e.g. PIXIE (Sleeman 1987)). These concepts will be elaborated further in Section 2.9.

As stated before, the term ILE (interactive learning environments) is generally used to denote all the kinds of learning systems based on active learning or constructivist learning theory. ILEs demand high-level context-sensitive guidance (Ramasundaram *et al.* 2005). Students are encouraged to be active learners (Bergin *et al.* 2003). The system and user are jointly expected to discover any knowledge in the environment (e.g. Exploring the Nardoo (Hedberg *et al.* 1995). In particular, CBL systems for programming that provide visualisation, algorithm animation, and other graphical facilities may be categorised as ILEs. A special kind of ILE called the *discovery world* will be discussed next.

2.4.1 Constructionism and Discovery Worlds

Constructionism is an important instructional theory under constructivism. This theory supports active learning as it demands creational activity. It is also goal driven. Seymour Papert (1972), a proponent of Constructionism, maintains:

> *"Constructionism--the N word as opposed to the V word--shares constructivism's connotation of learning as "building knowledge structures" irrespective of the circumstances of the learning. It then adds the idea that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it's a sand castle on the beach or a theory of the universe"* (Papert 1991, p. 1).

According to Constructionism theory, the following ten major factors affect a student's learning process: Learning by Doing, Learning How to Learn, Diversity of Personal Status, Learning Climate, Learner's Personality, Physical Intelligence, Self-Problem Solving, Intellectual Intelligence, Evaluative Feedback, and Learning from Tools and Equipment (Jitgarun 2004). Constructionism is the basis for a kind of ILE called *discovery worlds* that provide an effective environment for active learning by engaging students in creational activities (e.g. POLYGONS (McArthur *et al.* 1999)).

Problem Transformation is one of the constructionist instructional techniques that demands active learning and prerequisite knowledge. In this approach, learners are motivated to construct new models based on the given model. The transformation task

must be challenging and rewarding. The topic to be learned may be, for example, either of the modelling processes or the fundamental theory behind the models. Problem transformation will be further discussed in Chapter 4.

2.4.2 Bi-modus Learning Environment

Cognitivism supports the information processing model. Knowledge is stored in the system. Learners are individually guided in a pre-determined manner to acquire it. To give support tailored to the individual learner, the system keeps a model for each learner (Shute *et al* 1995). However, inevitably, this Learner model is not accurate. The system usually has a significant control over the learning process. On the other hand, in the constructivist view, the learning system should provide a suitable environment for the learner to construct their own knowledge. The system cannot provide individual support. The learner will have control over the learning process, or in other words, the learner is fully responsible for their own learning. In a practical approach, Intelligent Learning Environments are designed by incorporating features based on both cognitive and constructive theories (see Section 2.2.4). McArther *et al.* refer them as *mixed-initiative systems* with locally intelligent models, and conclude, "*Most effective learning environments include a mix of direct teaching, more passive support for learning together with substantial student choice* (McArthur *et al.* 1999, p. 58)". Examples of this approach include Intelligent Discovery Worlds (e.g. SMITHTOWN (Shute *et al.* 1990b)), Intelligent Programming Environments (e.g. INTELLITUTOR (Ueno 1994)), and Guided-Discovery Environments (e.g. (Elsom-Cook 1990)).

Those who favour the philosophy of situated cognition argue for a social and collaborative learning environment. In this view, the computer is considered merely as a tool (albeit a very important tool) to facilitate a learning environment (Jonassen 1995). The role of AI is considered to be marginal in the sense that it is required only to mediate any collaborative interactions (Paiva *et al.* 1995). Anchored Instruction is an instructional approach closely related to this philosophy (CTGV 1993). The communication capacity of computer networks, internet and intranets, are yet to be fully utilised to create effective Computer Supported Collaborative Learning systems.

Human beings are innately motivated to learn new everyday knowledge. However, for academic learning, motivation plays a vital role. In the next section some key motivational theories are outlined

2.5 Motivation and Learning

Motivation is defined as a process whereby goal-directed activity is instigated and sustained (Pintrich *et al.* 2002). The impact of motivation on learning has been well analysed in educational psychology, and later in cognitive science, disciplines. Early instruction design methodologies were founded primarily on noncognitive motivational theories (in which the learners are considered to be passive and controlled by the environment). In contrast, recent cognitive motivational theorists such as the proponents of self-efficacy theory propose that the learners take control over the environment (Ormrod 2003).

Motivations are usually labelled as either intrinsic or extrinsic. The formulators of Self-Determination Theory (Ryan *et al.* 2000), describe intrinsic and extrinsic motivations as, *"The term* extrinsic *motivation refers to the performance of an activity in order to attain some separable outcome and, thus, contrasts with* intrinsic *motivation which refers to doing an activity for the inherent satisfaction of the activity itself"* (Ryan *et al.* 2000, p. 71). Keller's (1983) ARCS model of motivation is basically for intrinsic motivation and suggests that the following aspects are to be considered important while designing instructional strategies: gaining Attention, demonstrating Relevance, building Confidence, and providing Satisfaction.

Organismic Integration Theory, one of the two sub theories within self-determination theory, details different forms of extrinsic motivation (ranked into four levels depending on their relative autonomy) and the contextual factors that either promote or hinder internalisation and integration of the regulation of behaviours (Ryan *et al.* 2000). Figure 2.3 illustrates the Organismic Integration Theory taxonomy of motivational types (the degree of self-determination increases from left to right). As the strength of self-determination increases, the positive impact of extrinsic motivation on the learning process also increases. Typically, in a learning environment the aim should be for the learner to achieve at least the third form of extrinsic motivation: identified regulation. In

the words of Ryan *et al*, *"A more autonomous or self regulated form of extrinsic motivation is regulation through identification. Identification reflects a conscious valuing of a behavioural goal or regulation, such that the action is accepted or owned as personally important"* (Ryan *et al.* 2000, p.72).

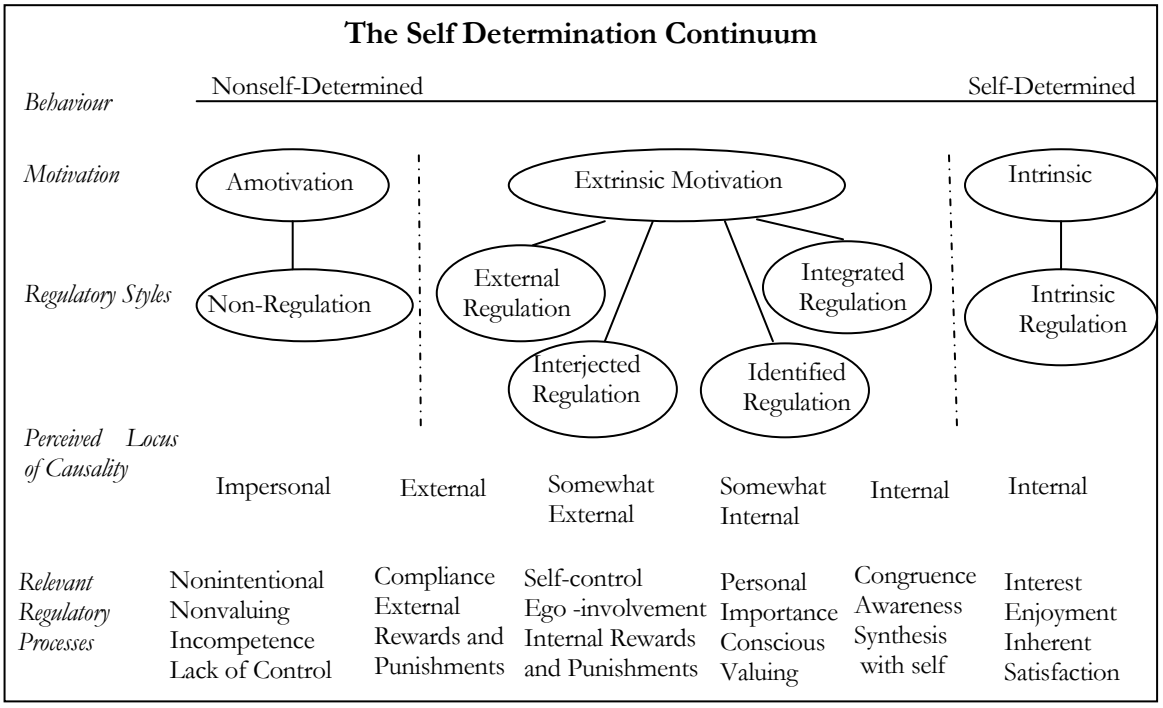


Figure 2.3 Types of Motivation (Ryan *et al*, 2000, p. 72)

A number of research results suggest that tertiary students or other mature students are less intrinsically motivated, but mostly extrinsically motivated, and particularly they give importance to the activities that are appropriate to their long-term goals. As the learner gets older, as Ryan *et al*. put it, *"the freedom to be intrinsically motivated is increasingly curtailed by social pressures …"* (Ryan *et al.* 2000, p. 71).

Learning and Motivational theories play an important role in CBL systems design. A number of instructional models and strategies based on major learning and motivational theories have been reported in the educational research literature (Reigeluth 1999). Some of them have been used successfully in CBL systems (e.g. Component Display Theory (Merrill 1983)). In the next section, one of the important instructional strategies based on the situated cognition theory, called the *Cognitive Apprenticeship Model* (Collins *et al.* 1990) is outlined.

2.6 Cognitive Apprenticeship Model

Borrowing the term from the traditional apprenticeship for learning practical skills, the aim in using the Cognitive Apprenticeship Model is to facilitate the learning of any discipline that involves complex cognitive processes. It is basically a learning paradigm and includes the following aspects: modelling, scaffolding and fading, coaching and feedback, articulating and reflecting, and exploring. The situated modelling suggests that the knowledge or skills are to be taught in an authentic context (the way it will be used in real life). The tacit and heuristic techniques that the experts usually use in the context are also to be taught. The learners will be supported in complex issues initially; and gradually the support will be reduced to empower the learner at the end. The role of teacher is changed to that of coach: the coach carefully watches the progress of a learner and provides appropriate support at the right time.

The learners are encouraged to explore alternative strategies and experience the effects in a relevant context. The global views should be presented before any specific feature. Meta cognitive activities are considered important in this model. Learners should be able to explain their own actions at present. They should also know what they have learned and how they have learned in the past. Eventually, through situated learning, Collins *et al*. argue, *"the students acquire knowledge in a dual form, both tied to contexts of its uses and independent of any particular context"* (Collins *et al.* 1990, p. 488). Scaffolding (and fading) is an important aspect of this model. In the next section, the notion of scaffolding will be discussed in more detail.

2.6.1 Scaffolding

Scaffolding is not a new idea in instructional design. In 1976, the term *scaffolding* was first used in education literature by Wood *et al*. (1976) to describe the strategies used by adults to train children, especially in language development. Originally, in one sense, the noun 'scaffold' is used to describe the structures used in building constructions to support the newly built portions until they are made strong enough to be self-supporting. Though the metaphor may lead to some serious misconceptions about the way scaffolding is actually practised in educational settings, it is being extensively used to describe the temporary support provided by educators to enable learners to build their own

understandings or skills. Though Vygotsky (1978) had not used the metaphor explicitly, his cognitive Zone of Proximal Development (ZPD) concept and Socio-Cultural Learning theory closely matches with the notion of scaffolding. ZPD is defined as *"the distance between the actual development level of the learner as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance or in collaboration with more capable peers"* (Vygotsky 1978, p. 86). According to these theories, learners need to be given necessary and sufficient support by their mentors or peers, based on their existing knowledge level, to build their own new skills or knowledge. The level of support varies among learners depending on their general capabilities and traits.

The scaffolding technique is often used for training in complex physical skills. The Cognitive Apprenticeship model uses scaffolding for learning complex cognitive tasks. In this model, to train a novice in a complex task, the coach initially completes the difficult processes and allows the trainee to concentrate on a portion of a task that can be comprehended within their working memory limits and ZPD. In this research, a four-phase instructional model, which is adapted from the Cognitive Apprenticeship model of Collins *et al*. (1990) is proposed for learning particularly complex domains with some prerequisite discipline (e.g. learning Object-Z notation with UML as a prerequisite).

The rest of the chapter outlines some pertinent issues related to CBL system design and development that are relevant to this research. Firstly, authoring systems are outlined. The key pedagogical decision making actions such as feedback selection and curriculum advancing are discussed next. Thereafter, Learner models are discussed in detail. Finally, a discussion is included on using a particular testing format (Multiple Question Test) for formative assessment.

2.7 Authoring Systems

Thousands of CBL systems have been developed for different disciplines. Still, there is no easy way to create one for a new subject domain. Each project needs to be started from scratch. The next section discusses this issue in detail. Creating a new CBL system is a time consuming and costly task (Murray 1999). It would be very useful if there were efficient and easy to use (for non programmers) authoring tools available to create CBL

systems. In fact, many such attempts have been reported. For example, Authoring languages (e.g. ADAPT for TICCIT (Faust 1974)), Unintelligent Authoring Systems (PLATO (Bitzer *et al.* 1961)), Multi-media Authoring Tools (with scripting languages e.g. Authorware (Macromedia 2001)), Intelligent Authoring Shells (e.g. ITSSEL (Tekinerdogan *et al.* 1995)), Visual Authoring Languages (e.g. CALVIN (Bell *et al.* 1993)), Intelligent Authoring Systems (e.g. IDE (Russell 1988)), Specific-type Intelligent Authoring Systems (e.g. RIDES (Munro *et al.* 1997)), and Meta-authoring Tools (e.g. Eon (Murray 1999)). Murray *et al.* (2003) in their influential book *Tools for Advanced Technology Learning Environments* discuss the state of this art in detail. They also suggest several tradeoffs in designing such tools.

Based on his experience of developing ITSs for teaching ADA, Mark Miller (1978) advised potential developers to create a relevant authoring tool first before designing any ITS. Psotka *et al*. (1988) concluded that although this is a good advice, it is not practical: they did not believe that authoring tools for PCs would be available in the near future. A decade later, Tom Murray (1999) asserted that ITS authoring tools are still research vehicles, and have not yet been made robust enough to be placed in production contexts. Recently, the authoring environment for REDEEM has been claimed to be a success after a review of its five year programme in real authoring tasks (Ainsworth *et al.* 2006). However, REDEEM is, the authors claim, suitable only for simple learning environments.

Important elements of any teaching activity are selecting suitable learning material and providing appropriate feedback at the right time. In the next section, these issues will be discussed in detail.

2.8 Pedagogical Action Selection

A typical adaptive system for procedural type domains usually performs four important actions: diagnosis, feedback selection, prediction, and curriculum sequencing. The diagnosis (and later prediction also) is usually based on incomplete and noisy data. Consequently, PAS (Pedagogical Action Selection) (Mayo 2001) is made using those unreliable diagnosis and error-prone predictions. If inappropriate feedback or an untimely sequence of lessons (e.g. due to a poor scaffolding process) is repeatedly given to the

learners they may become confused or irritated, and ultimately their motivation could be affected.

2.8.1 Curriculum Sequencing

The two key features of intelligent models in CBL systems are Curriculum Sequencing support and Interactive Problem Solving support (Weber *et al.* 1997). Brusilovsky *et al*. (2003) define the term Curriculum Sequencing (or Course Sequencing) as the act of generating an individualized course for each student by dynamically selecting the most appropriate pedagogical operation, usually after a learner's interaction (or in certain time interval). The possible pedagogical actions include presenting core material, a worked example, or a demonstration; asking questions or seeking verifications; providing suggestions and solutions; and conducting formative or subjective assessments. Feedback, due to its importance in learning, will be considered separately. The term *task sequencing* is traditionally used for selecting suitable test items (Brusilovsky *et al.* 2003).

Brusilovsky *et al.* (2003) discuss three approaches for Course Sequencing in large web applications. Usually, to support course sequencing, the subject knowledge will be organised as a network of concepts, and the Learner model will be given as a weighted overlay of this network. The notion of *interactive problem solving support* may be considered as Curriculum Sequencing at a micro level. For effective interactive problem support the domain model should be organised at a fine-grained level.

2.8.2 Feedback

To err is human. Feedback plays a key role in learning. Research into feedback has a long history. Kulhavy *et al*. (1990) discuss the effectiveness of different levels of feedback in different situations (Table 2.1). The term *response-certitude*, which was coined by Kulhavy *et al*. (1990), denotes the metacognitive estimate of the learner's understanding of the material combined with their understanding of the question and response possibilities (in other words, it denotes the confidence level of the learner in their selected responses). Kulhavy *et al*. (1990) argue that when a learner has a high level of confidence in their selected answer, and if the answer is also correct, just a verification is sufficient as the feedback. On the other hand, if the given test item is very difficult for a learner, s/he may still need some informative feedback even if s/he answered it correctly (it may have

been a lucky guess) (Mason *et al.* 1999)., the learner may be tested at the same level to ensure that the previous success was not merely a lucky guess.

Type	Verify	Answer Given	Response Analysed	Post Action
No feedback	not individually	no	no	Total marks
Knowledge-of-Response	yes	no	no	Each item
Answer-Until-Correct	yes	Later no	no	Repeats
Knowledge-of-Correct Response	yes	no	no	Direct to the source
Topic- Contingent	yes	yes	no	Why answer is correct?
Response-Contingent	yes	yes	yes	Analyse both
Bug-related	yes	no	yes	Bug identified, analysed, and Give directions
Attribute-isolation	yes	yes	no	Key concepts Past attempts Counter example

Table 2.1 Feedback Types - based on (Mason *et al.* 1999)

Mason *et al.* (1999) provide a flowchart (Figure 2.4) for decision making to obtain the most suitable feedback in CBL systems based on different factors such as the student's past achievement and current task level. Mory (1996) states that delaying the feedback, especially for MC tests, significantly impacts the learning process positively. Delaying feedback may be useful while guiding learners to enter deadlocks in order to give them an opportunity to work out their own misconceptions.

Kulik *et al.*, in their classic paper *Timing of Feedback*, conclude that "*the delayed feedback appears to help learning only in special experimental situations and that more typically to delay feedback is to hinder learning"* (Kulik *et al.* 1988, p. 94). In the context of computer-based instruction, Azevedo *et al.* assert "*immediate delivery of a feedback message provides the best instructional advantage to the student*" (Azevedo *et al.* 1995, p. 122). Being part of the scaffolding process, the immediate feedback option is preferred in this research. More discussion on feedback will be given in Chapter 5.

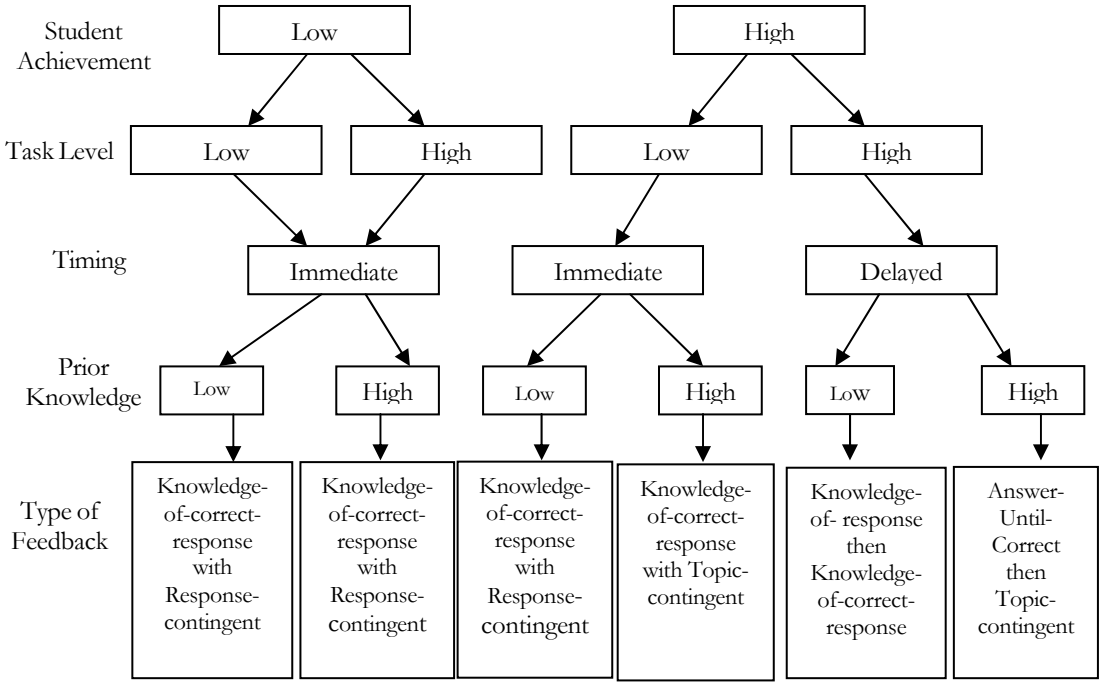


Figure 2.4 A Decision-Making Framework for Feedback (Mason *et al.* 1999)

2.9 Learner models

The Learner or Student model is an abstract representation of a student in the current learning context. Basically, it is a collection of the system's beliefs about a learner (Holt *et al.* 1993). The learner may provide some detail explicitly, and/or the system may infer some information based on the learner's interactions. The Learner model is essential for providing motivational and adaptable learning environments tailored to a variety of learners (Greer *et al.* 1994). An ideal model may include the learner's prerequisite, prior, and current knowledge states, and various relevant attributes of the learner such as preferred learning styles and traits. Especially, as the learning progresses, the learner's current knowledge states will be updated to reflect the change. A human learner thinks dynamically and in an unpredictable manner. Even human teachers may often need to predict the learner's knowledge state based only on their behaviour. However, the current state of human computer interaction is not capable of interpreting the learner's behaviour properly and completely. Therefore, designing an efficient Learner model is significantly challenging, if not impossible.

Holt *et al*. (1993) classify the early Learner models into three types: overlay, differential and perturbation models. In an *overlay* model, the domain model contains only subject knowledge, and the learner's knowledge is always considered as a subset of the domain model. A simple overlay model may record whether a knowledge item is learned or not. This feature helps the system to prevent the learner from unnecessary revisiting (e.g. BIP (Barr *et al.* 1975)). A variety of ITSs from Anderson's camp have been developed based on ACT* theory which uses the model tracing approach for student behaviour modelling (Anderson *et al.* 1985). Model tracing is a kind of overlay model. This kind of system is based on a well-defined curriculum that has been arranged to promote knowledge/skill acquisition. A main drawback of model tracing is that it is very restrictive and will not accept student errors. It works very well in procedural skill acquisition, well structured or rule-based domains.

A subtle variation is the *weighted overlay* model, where a weight is assigned to each knowledge item to record how well the item has been learned. If the domain model is a complex curriculum tree such as an enhanced network of knowledge items with additional information such as conditional relationships and related probabilities, a weighted overlay model may be used to make various predictions by attaching suitable logic. With the assistance of rich domain models, these Learner models can provide interactive problem solving help (Corbett *et al.* 1993; Conati *et al.* 2002).

In the *differential* model, the existing domain model is considered as being incomplete. The learner may demonstrate some knowledge different from the domain model (e.g. WEST (Burton 1982)), whereas, the perturbation or buggy model augments the subject knowledge with potential bugs in its domain model (DEBUGGY (Burton 1982)). This is also called an *impasse-driven* model. An impasse is a situation where the model has insufficient knowledge to determine how to proceed (VanLehn 1982). A notable deviation of this is the mal-rule approach (Sleeman 1987), where mal-rules are inferred or abstracted from the basic principles and bugs (VanLehn 1982). Bugs are, in general, simple working errors whereas mal-rules are related to deep misconceptions. The misconceptions may be incorporated in three ways: the indexed list of misconceptions may be created beforehand (e.g. PROUST (Johnson *et al.* 1985)), or the potential misconceptions may be reconstructed case by case (e.g. PIXIE (Sleeman 1987)), or all

possible variations of misconceptions may be generated (Repair theory (Brown *et al.* 1980)).

2.9.1 Uncertainty in Learner modelling

A typical Learner Model performs four actions: diagnosis (knowledge tracing or plan recognition), feedback selection, predicting and model updating, and curriculum advancing. The term *knowledge tracing* (Corbett *et al.* 1992; Conati *et al.* 2002) refers to the process of inferring the learner's knowledge state from their behaviour. Plan recognition is slightly different from the diagnosis task. Diagnosis is an attempt to attribute meaning to student behaviour during the performance of a task, or after it has been completed, whereas *plan recognition* describes an attempt to identify the plan or goal of the learner before they complete a task (Greer *et al.* 1995). A fine-grained domain model is necessary for micro level diagnosis in order to provide interactive problem solving help (while a learner is working on a solution) in procedural type learning domains.

If a CBL system makes critical assumptions based on an inaccurate Learner model and vague observations, it cannot carry out perfect diagnoses. Therefore, the potential remedial actions which are usually based on these inaccurate diagnoses could be inappropriate. Later, if the Learner model is updated in the light of having been given useless feedback, the prediction could be wrong. Furthermore, the next lesson, if selected using a new predicted Learner Model, could be untimely. Iteratively, if this process continues, the errors could accumulate, and eventually the system might behave erratically (Mayo 2001). The resultant Learner model could significantly deviate from the real user (Figure 2.5). Eventually, this would adversely affect the trustworthiness of the system and could seriously impact the motivation of the learner.

Villano (1992) suggests a range of strategies to use the Bayesian Belief Network (BBN) for typical pedagogical actions such as item selection (the best next question), updating, curriculum advancement, hint-selection (during problem solving) and feedback (after problem solving). Basically, he uses prerequisite links between various test-items using AND/OR graphs. In the Corbett *et al.* (1992) scheme, the Learner model is a set of LISP programming rules with a tag to denote whether a rule is in the "learned" or "unlearned"

state (with no forgetting). For each rule, the probability that a rule is in the learned state will be associated for a learner. Though it is not explicitly stated, the Dynamic Bayesian Network rules (Russell *et al.* 2003) are used to update this probability with relevant interaction. An important characteristic of Corbett *et al*'s (*ibid.*) model is that the programming rules are considered independent.

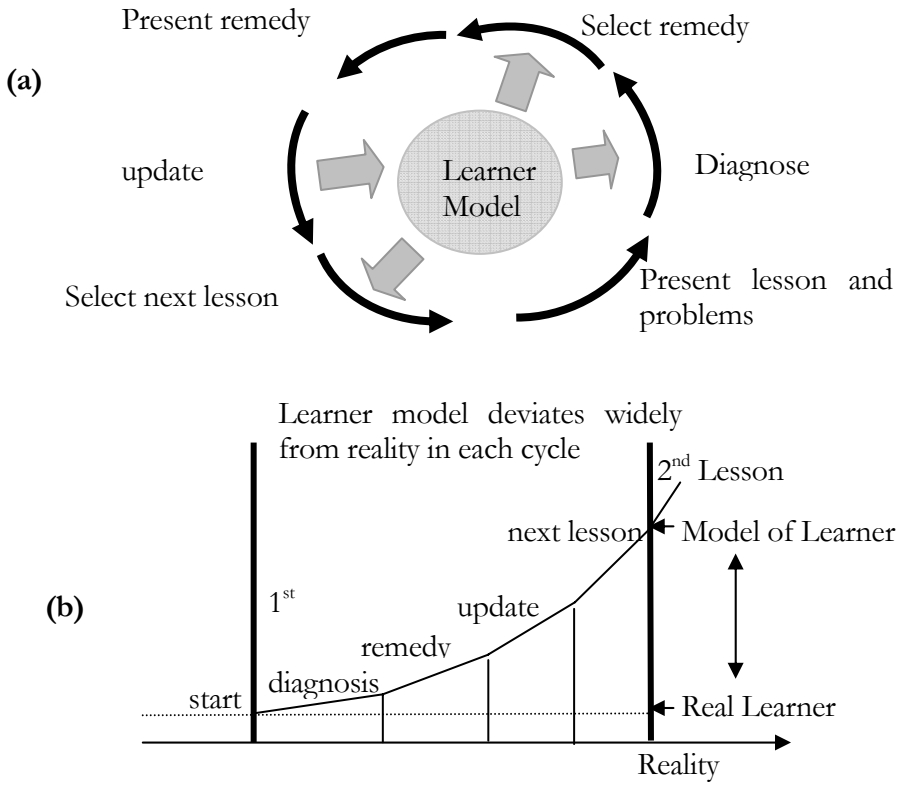


Figure 2.5 (a) Learner Model Usage (Mayo *et al.* 2001) **(b)** Uncertainty Escalation

Reye *et al*. (1995) suggest the usage of statistical decision theory for post-diagnostic pedagogical actions. Collins *et al*. (1996) use BBN for adaptive testing based on granularity hierarchies (McCalla *et al.* 1994) and then use a utility function based on statistical decision theory for item selection (the best next question). Mayo *et al*. (2001) also use utility functions with BBN for curriculum advancing in their systems called CAPIT. As Mayo *et al*. put it:

> *"It* [pedagogical module of CAPIT] *performs two significant PAS tasks. Firstly, given the violated constraints, it selects the single violated constraint about which feedback should be given, secondly when Pick Another Problem is clicked, or when the student solves the current problem, the pedagogical module selects the most appropriate next problem for the student"* (Mayo *et al.* 2001, p.14)**.**

CAPIT is based on constraint-based modelling (Ohlsson 1994), a successful variation of overlay models (Mitrovic 2005). The structure and prior probabilities of the network in CAPIT are selected using machine learning techniques. Murray *et al.* (2000) also used utility functions for curriculum advancing in their systems called DT Tutor. The backbone of DT Tutor is the Dynamic Decision Network, which is essentially a Dynamic Bayesian Network with decision and utility nodes. It also accounts for the student's focus of attention and emotional states. Stern *et al.* (1999) use machine learning techniques in MANIC to improve the prediction using BBN in user modelling. Other than DT Tutor, Conati *et al.* (2002) describe a CBL system for Newtonian physics, ANDES, that uses BBN for three key tasks: knowledge tracing-inferring from student's actions what the student knows, plan recognition- why the student did something, and prediction- what will the student do next. To enable micro level diagnosis, the knowledge in ANDES is represented at a fine-grained level using AND/OR graphs, but, unlike Villano (1992), ANDES uses aggregation links (instead of prerequisite links).

Most of the systems discussed above use belief propagation when external evidence becomes available. It is computationally a very expensive task. However, Murray *et al.* are more optimistic *"Unfortunately, belief network inference is still NP-hard in the worst case. However, many stochastic sampling algorithms have an 'any time property' that allows an approximate result to be obtained at any point in the computation"* ((Murray *et al.* 2000, p. 154).

Hawkes *et al.* (1990) first proposed fuzzy theory (Zadeh 1973; Hopgood 2000) for Learner modelling in the system called TAPS. The following usages of this model other than diagnosing are enthusiastically discussed; selecting instructional routine, intervention frequency, and mentoring type. The learner's emotional states were also included in the fuzzy rules in their model. Later, Katz *et al*. (1992) use fuzzy theory in SHERLOCK-II for knowledge tracing and diagnosis. They use the term *knowledge state* for a possible level of competence of a trainee on a skill or concept. In order to apply fuzzy rules, they use a kind of point-score scheme to update the strength of knowledge states. Though fuzzy logic is easy to understand and manipulate, it is not used to its full potential for uncertainty management in Learner models. Some other approaches for uncertainty handling in user models are discussed in Jameson (1996).

Maintaining optimal Learner models is challenging, mostly due to the uncertainty associated with assigning credits between various probable causes based on observed evidence. One of the solutions suggested is opening the Learner model to the users. In the next section this issue is discussed in detail.

2.9.2 Opening Learner models

Opening Learner models is not a new idea. In particular, the act of opening up an adaptable Learner model to learners, peers, and mentors has been given much attention in the past. Basically, a Learner model is an abstraction of the beliefs of the system based on certain characteristics of a learner. Opening the Learner model is an act of providing the information kept in a Learner model (and how it would be utilised) in a usable format for other agents (Paiva *et al.* 1995). Initially, the idea of opening Learner models to the learners was suggested by Self (1990), in order to share the burden of designing and maintaining adaptive Learner Models. However, the current trend in the research towards opening Learner Models reflects contemporary learning theories (Kay *et al.* 2005). Constructivist learning theories suggest that the instructional design should include strategies for enhancing collaborative learning and meta-cognitive activities such as reflection. Furthermore, opening up limited parts of Learner Models to peers, and mentors has also been given much attention in recent research. This helps learners not only to estimate their own abilities but also compare them with those of their peers.

The following are some of the advantages claimed for opening Learner Models (Self 1990; Kay 2000; Mitrovic *et al.* 2002; Bull 2004):

- Reduces the burden of Learner models significantly.

  Maintaining optimal Learner Models is very difficult, if not impossible. Learners are allowed to change the measurements kept by the system. The role is now shared between the system and the learner. A possible misinterpretation by the system may be corrected by the learner in the early stages.

- Improves learners' meta-cognitive activities.

  Learners can view what they learned and how much they have learned. They can backtrack to discover how they learned and how they responded in various system states and for different system behaviour.

Learners could:

- reflect on their own knowledge and skills
- understand their weak areas and misconception
- estimate their own general qualities and capabilities
- identify their learning styles, preferences, etc.
- take control over their learning process

- Maintains learners' intrinsic motivation

  Learners can evaluate their performance over time. They can also view and compare peers' models (limited only by law and ethics). Cognitive motivational theories - such as self-efficacy theory - suggest that the learners' intrinsic motivation is increased – especially by mastery and vicarious experience (Pajares, 2002).

- Keeps Learner- Mentor link active

  Mentors can view the learners' models individually and collectively. Learners could comment on their own Learner model and others' expectations. The mentor can motivate learners in different ways depending on their individual needs.

- Abides by the privacy laws of some countries.

  Some countries have extensive privacy laws, which demand that all the information kept by a system about a person should be revealed to that person, if requested.

Paiva *et al.* (1995) describe an open system called TAGUAS and mention several other related pieces of research (e.g. Martin *et al* (1993)). TAGUAS separates the Learner model and its functionality from the underlying applications. Mitrovic *et al.*(2002) discuss some other research, and particularly, presented the encouraging results of the evaluation of one of their systems (SQL Tutor). Bull (2004) discuss a variety of strategies and interfaces used in opening Learner models with illustrations.

Importantly, in other research carried out in the ARIES laboratory (Zapata-Rivera *et al.* 2004), a graphical tool called *VisiMod* (Visualization of Bayesian Learner models) is used to assist the learners and teachers to inspect and modify a complex Learner model that uses a Bayesian Network. The parameters are mapped to fuzzy-like attributes (good, very-good, expert) and can be modified simply by using sliders. Two artificial agents are also

used to guide the learners through the interaction. Extensive evaluations based on usability and explorative studies are also included in this paper. Since a fuzzy Learner model can be designed based on simple rules that resemble typical human teachers' decision making processes, comparatively, opening a fuzzy Learner model would be easier than its Bayesian counterpart.

Formative assessment is very important part in the scaffolding process. Multiple Choice (MC) tests may be used for formative assessment. In the next section, some key issues behind using MC tests in assessments will be discussed.

2.10 Assessment in CBL systems - The Role of MC test

CBL systems usually include formative assessment units in order to estimate the level of understanding of a learner on a topic. Particularly for automatic formative assessments, MC tests are preferred as they are easy to administer and mark. MC test could also cover a wide range of topics. Obviously, it is difficult to design software that automatically marks essay type questions. However, MC tests are not sufficient to assess the level of understanding precisely. The interface bandwidth is limited- it only allows users to select one or more answers. The following paragraphs discuss such criticisms and potential solutions in detail.

An MC test consists of three parts, a question (stem), the correct answer (key) and other options (distracters). The potential of MC tests in educational testing is widely discussed in the educational psychology and measurement literatures. In the late 1990s researchers in educational assessment regarded them as a type of objective test that is used for measuring knowledge and skills (Ben-Simon *et al.* 1997). Recently, Simkin *et al*. (2005) have listed more than 15 advantages of using them. However, there are three main criticisms which are discussed in the educational psychology literature on using them for assessment (Simkin *et al.* 2005):

- They are not suitable for assessing higher level cognitive abilities such as synthesising or evaluating (in Bloom's taxonomy (1956)).
- They are not reliable, as guessing may still play a role.
- They cannot measure partial knowledge.

Each criticism will be considered in turn. The first one claims that MC tests can assess only recognising or recalling ability, rather than applying, analysing, synthesising or evaluating ability, where the latter cognitive tasks are more difficult than the first based on Bloom's taxonomy (Bloom 1956). A related criticism is that MC tests lack structural fidelity: *structural fidelity* is defined as "*the congruence between performance called upon by the test and proficient performance in the referent domain"* (Simkin *et al.* 2005, p. 77). Although some studies by educational psychologists have demonstrated that carefully selected MC tests could be efficient as Constructed Response (CR) tests (such as essay-type or problem solving tests) in testing higher level abilities MC tests are considered inefficient (*ibid.*). In a recent study in the domain of computer programming languages the researchers assert the above claim. Simkin *et al*. state, *"Despite research from educational psychology demonstrating the potential for MC tests to measure the same levels of students' mastery as CR tests, recent studies in specific domain find imperfect relationship between two performance measures"* (Simkin *et al.* 2005, p. 73). They also maintain that in programming language instruction and similar domains CR tests measure the student's ability to solve real world problems better than do MC tests.

The second criticism of MC tests is the fact that just because a student has answered a test question correctly does not imply that s/he has mastered the concept. The result of an MC test may not be reliable as the "testwiseness" (clever format-specific strategies) might also play a role (Simkin *et al.* 2005). There have been a number of attempts to try to alleviate the effect of lucky guesses in conventional (number-correct) MC tests. Normalisation (Bush 2001) is one of the easiest correction-for-guess marking methods used in practice to smooth this effect. In normalisation, the total mark is normalised. In one-in-four MC tests (four options with one correct answer), the probability of a correct guess is ¼. Therefore, for n test items with x correct, the correction is $(x-n/4)*4/3$. In correction-for-guess, a penalty of 1/3 is imposed for each incorrect answer. Other than these methods, allowing multiple correct answers in a test also reduces the effect of guessing considerably (Bush, 2001). This form of correction-for-guess marking methods, however, has been severely criticised by researchers as it penalises the candidates whose unsuccessful guesses were made by partial knowledge (informed guesses rather than blind ones). In the next section a number of research attempts that address the third criticism by giving reasonable credit for partial knowledge in MC tests are discussed.

2.10.1 Rewarding Partial Knowledge

Several scoring methods have been proposed to reward the guesses made using partial knowledge. Ben-Simon (1997) gives a comparative study between different methods that account for partial knowledge. In the 1950s, Coombs *et al*. (1956) devised a schema called *Elimination Testing procedure*. In this procedure, students need to mark all the incorrect answers. One point is given if a wrong answer is marked as wrong (but three points will be deducted if the right answer is marked as wrong). Even after five decades, the Coombs method is considered to be more effective than many recent such methods (Bradbard *et al.* 2004). Bush (2001) explains his "liberal test" method (a sort of subset selection method) and claims it is better than many other methods. In the subset-selection-method, the candidate is asked to specify a subset that includes the correct answer. If the subset is a singleton set with the correct answer 3 points are awarded. The mark decreases, if the correct answer is included, with the size of the subset. A variation of this method is *series selection*, where the *order of preference* is to be given explicitly. Higher marks will be awarded when a correct answer is at the top (*ibid.*).

In line with this research, Gardner-Medwin's (1995) Confidence-Based Marking, which has been used for the last 10 years for formative assessment (and for four years for summative assessment) in a medical course at University College, London, is worth discussing further. Gardner-Medwin states, *"confidence-based marking (CBM) has been known for many years to stimulate reflection and constructive thinking by students and improve both the reliability and validity of exam data in measuring partial knowledge"* (Gardner-Medwin 2005, p. 1).

He further explain how CBM works,

> *"Our system employs three confidence levels 1-2-3. Students are asked to rate their confidence after each time they have answered a question (True/False, MCQ, numeric, or open text) that will be marked categorically right or wrong. With low confidence they receive just 1 mark if correct and no penalty if their answer is wrong. At levels 2, 3 they receive accordingly 2 or 3 marks if correct, but increasing penalties (normally -2 and -6 marks) if wrong"* (Gardner-Medwin 2005, p. 1).

If the distracters on an MC test are related to potential misconceptions, identifying the knowledge state of a student minutely on each item is crucial in order to provide rich item-specific feedback. However, Gardner-Medwin's method requires the student to state their confidence on one selected answer only. The student's knowledge state on other answers will not be explicitly identified.

2.11 Summary

Learning theories play an important role in CBL systems research. A number of different architectures for CBL systems have been designed and tested. Early CAI systems were based on behaviourist theory. ITSs are based on cognitive theory - they lack learner control. Whereas, ILEs are based on constructive theory - they generally lack learner support. Systems in the middle ground could provide active learning with guided support using locally intelligent Learner models. There are four different epistemological positions in constructivism. At one extreme, the cognitive constructivist argues that there is a single true external reality and knowledge is individually constructed. At the other extreme, the social constructivism (situated cognition) claim that there are multiple realities and knowledge is constructed as a social process.

In addition to learning theories, motivational theories also significantly influence CBL system design. First of all, a system should be able to motivate learners to use it and it should continuously maintain their motivation at high. Mature students are usually extrinsically motivated and they give priority to long-term goals.

Constructionism is an important instructional theory founded on constructivist learning theory. Under this theory, learners are encouraged to actively engage in some creational goal-driven activity in order to build related knowledge structures. Problem Transformation is one of the constructionist instructional techniques that demands active learning. In this approach, learners are motivated to construct new models based on the given model. The transformation task must be challenging and rewarding. Scaffolding may be used if the transformation processes are complex. Problem transformation will be discussed in detail in Chapter 4.

The cognitive apprentice model is an important instructional model based on situated cognition. This model is suitable for learning disciplines that involve complex procedural tasks. Scaffolding is the key strategy used in this model. Initially, the learners will be supported in complex issues, and gradually the support will be withdrawn. Eventually, learners will be able to do the complex tasks without any support. Coaching rather than teaching is involved in learning process. Metacognitive activities such as articulation and reflection are two key components of this model. The apprentice model, however, needs to be modified to cater for academic knowledge acquisition.

Selecting suitable pedagogical actions such as course sequencing and feedback are important activities of CBL systems. Intelligent Learner models may provide individualised support to the learners. Learner models should keep relevant past interactions and it should be able to predict the current knowledge level based on the recent interactions. This estimate can be used by the teacher model to provide adaptive assistant. Nevertheless, designing an optimal Learner model is difficult. Since the learner input is limited, uncertainty is inevitable. Handling uncertainty in Learner models is a challenging task. A Bayesian Network is the main paradigm usually used for uncertainty management in Learner models. Fuzzy logic is also used for this purpose but to a lesser degree. The act of opening Learner model has many advantages. It is hard to open Bayesian models. Since fuzzy rules may resemble typical human teachers thinking processes, opening fuzzy models may be easy.

Finally, formative assessment is essential in CBL systems in order to estimate learners' current level of knowledge. MC tests are widely used this purpose. They are easy to administer in a computer environment. They are also easy to mark automatically. However, MC test interface bandwidth is limited, and therefore, it is not sufficient to measure the true status of a student. The confidence-based multiple-choice schema provides an elegant way of measuring partial understanding. Nevertheless, it is not capable of measuring the knowledge state of a student on each choice of a MC test. This requirement is, however, essential in order to provide item-specific feedback.

Chapter 3

CBL Systems for Programming and Formal Methods

3.1 Introduction

Learning formal specification and computer programming, two important disciplines in software engineering, are usually considered challenging by undergraduate students. In this research, Object-Z, an object oriented extension of the formal specification language Z is used as the exemplar subject domain. The ultimate aim of any formal or informal specification is creating relevant programs. Some formal specifications may be automatically (or with limited human interaction) refined to computer programs. Compared to the research related to CBL systems for formal specification there have been plenty of research activities reported for CBL systems for programming. Being formal notations, both learning computer programming and learning formal methods have many common features. In order to get a wider picture of the exemplar domain, firstly, a limited survey on CBL systems for programming is included in Section 3.2.

In Section 3.3, an overview of formal specification notations and tools is given (Object-Z notation will be discussed in detail in Chapter 4). Some key issues related to teaching formal notation are also discussed. Section 3.4 outlines past research on designing teaching systems for learning formal methods. To the knowledge of the author, there has been nothing reported in the open literature regarding any learning system for Object-Z. However, there is some research available for teaching systems (or tools) for some other non-object oriented formal methods. Finally, this chapter outlines some of the research on computer based tools and teaching systems for formal methods.

3.2 Teaching Systems for Programming

3.2.1 Overview

One of the main reasons for the ineffectiveness of software systems to harness the capability of hardware technology is the lack of productivity of computer programmers (Sommerville 2001). Productivity, however, could be increased using effective software tools. Within the last four decades, there have been hundreds of CBL systems reported in the literature for almost all the popular computer programming languages of different paradigms (du Boulay *et al.* 1987; DeWolf *et al.* 1992; Deek *et al.* 1998). The majority of programming tutors – exceptions include the LISP tutor (Anderson *et al.* 1985) – address elementary problems rather than real programming tasks. Meanwhile, there are many CBL systems for programming that have been developed for the internet or intranet (Deek *et al.* 2001). Nearly all such CBL systems deal only with the particular language (or a paradigm) they are designed for, and support only a particular aspect of programming tasks (some provide help in syntax editing, or debugging, or designing, or planning).

Nevertheless, learning programming is challenging and moreover, the creation of software applications that could be used to learn knowledge and skills necessary for computer programming is also challenging. Although there have been many attempts made in this regard, only a few of these applications are being seriously used as a learning aid in real education.

3.2.2 Challenges and Drawbacks

Several researchers outline various reasons for the difficulties associated with learning programming (Shneiderman 1977; Soloway *et al.* 1982; Deek *et al.* 1998). The most critical errors made by novice programmers seem to be due to an inability to:

- abstract a real-world problem to plan a solution.
- devise a solution for a given computer environment
- understand the semantics (iteration, recursion, polymorphism etc.).
- grasp the complex syntax of the given environment.

For several years, there has been concentrated research aimed at designing efficient CBL systems to ease the difficulty in learning programming. However, most of those systems have not had any commercial success. A decade ago, Brusilovsky (1995) mentioned that the "real world" situation, relating to classroom application of the CBL systems for programming at that time, was no better than it was 20 years before. It seems that this odd truth still prevails. There are several reasons cited by different researchers for this situation (du Boulay 1992; Deek *et al.* 1998). Collectively, the important deficiencies of CBL systems for programming are listed below.

- Deal with toy problems only – insufficient knowledge base
- Less interactivity – not motivating
- Do not address the planning deficiency of a beginner programmer at all
- Confined to a particular programming language or a paradigm
- Not designed according to Instruction/Curriculum Design techniques, and not evaluated seriously

3.2.3 CBL Systems for Programming

There have been three full-scale surveys carried out on CBL applied to programming (du Boulay *et al.* 1987; DeWolf *et al.* 1992; Deek *et al.* 1998). For this research, it is sufficient to overview those surveys, and some other key systems reported after the recent survey. du Boulay *et al.*'s work (1987), the first of this type, is very interesting and valuable, and covers almost all the important early CBL systems for programming. They classify the systems based on two factors: the educational role they play and the type of knowledge they deploy, and they group them into three main classes: Tutors (and coaches), Bug Finders, and Support Environments. They characterise 'Tutors' based on Hartley's definition (see Section 2.3.2 for more detail (Hartley *et al.* 1973)), and they differentiate 'Bug Finders' as a group of systems that can find a bug in a program, analyse and advise (if they are asked to do so). The 'Support Environment' is described as *"…usually intelligently designed (rather than inherently intelligent) to provide easy-to-use tools for different part of the programming process…"* (du Boulay 1988, p. 1).

After five years, the next survey was conducted by DeWolf *et al.* (1992). They classify the systems into two dimensions: competence in domain knowledge and competence in teaching capability. In particular, under the domain knowledge dimension they divide

systems into three groups, depending on the area of the system's knowledge: program constructions, program verification, or neither of them. Each of these divisions closely matches with Tutors, Bug Finders and Support Environments, respectively. For example, the systems which have knowledge in program constructions may provide interactive help. If they can also address individual students they will become du Boulay's (Intelligent) Tutors. Knowledge in program verification is essential for debugging. For support environments, none of this knowledge is essential.

Deek (1998) divides the systems into four categories. His classification schema also matches closely with that of du Boulay's but with slightly different labels. In addition to the three groups mentioned before, he includes a new category called the intelligent programming environment. This architecture has now become prominent in CBL research, since it includes advantages of both artificial intelligence and constructive learning theories. Deek (1998) and du Boulay (1988) warn that their classifications do not follow any hard-and-fast rules, and therefore, should not be considered as a clear-cut division.

There have been some research attempts to migrate existing learning systems to the web (Brusilovsky *et al.* 2003). Since the millennium, there has been considerable research reported in the literature on CBL systems for programming. Most of these studies concentrate on designing learning systems for internet (or intranet) platforms (see (Deek *et al.* 2001) for detailed discussions on some earlier web-based systems for programming).

Intelligent Tutors and Bug Finders

Tutors have some capacity to monitor students' actions and they can also teach, instruct, or coach appropriately, whereas Bug Finders can only locate logical errors in the students' completed solutions and provide appropriate feedback. LISPITS (Anderson *et al.* 1984) and PROUST (Johnson *et al.* 1985) are classical examples of a Tutor and a Bug Finder, respectively. Some recent examples include PROPL (Lane 2003) of a Tutor, and (Xu *et al.* 2003) of a Bug Finder.

LISPITS is based on ACT* theory (Anderson 1983). Production rules are included to represent the potential solutions as well as deviations. The model tracing approach is employed for instruction. Each new symbol the student enters will be parsed. If the system identifies that the student's solution path is correct it will allow them to proceed. Otherwise, if the system identifies it as a known deviation, it will provide appropriate feedback. For any unknown behaviour, the system will provide suggestions for alternative actions. Similar to Formalizer (Flynn *et al.* 1989), the editor includes place holders for the students to fill; and therefore, they are freed from worrying about the syntax of the language. There is a computational advantage by the restriction of the potential paths and the provision of place holders. This approach may also be helpful for the students at their initial stages of learning. For advanced students, however, this approach may be frustrating due to its lack of flexibility.

PROUST can successfully detect a variety of logical errors (Johnson *et al.* 1985). However, unlike LISPITS, it is not pro-active; it can debug programs only after the task is completed. It includes a rich set of empirical data based on both correct and incorrect answers for certain programming problems from potential novice learners. The program specification is given as a sequence of goals. Each goal can be associated with a sequence of plans. The students' solutions are dissected and their intentions are identified as a structure of goals and plans. PROUST is powerful in a limited range of problems, but it can however, be easily fooled and it cannot work interactively (du Boulay 1988).

PROPL (Lane 2003) is a dialogue-based coach for program planning. Before attempting to write a program, similar to BRIDGE (Bonar *et al.* 1986), learners are asked to identify programming goals and the best way to achieve them. The design ideas for a programming problem are elicited from the student through a natural language dialogue (Shapiro (1982) uses dialogue strategy in the debugging stage). This approach has been tested with sixteen subjects, and the authors claim it is a success where the coached students performed better than the control group.

Xu *et al.* (2003) describe the transformation-based strategy used in their 'Bug Finder'. The errors in students' programs are diagnosed by incorporating control-flow analysis and data-flow analysis in program analysis. Similar to LAURA (Adam *et al.* 1980), another classical tutor, the student program and a specimen program are standardized and then

compared at the semantic level. The authors claim that their approach was tested on more than five hundred real student programs (for nine different programming tasks), and the results were encouraging.

Visual (or Textual) Support Environments

Tutors and bug finders do not exploit the exclusive features of the programming domain, and concentrate instead on tutoring the problem solving skills. A learning system for programming should help the learner to create the appropriate mental model of the dynamic behaviour of the program and underlying notional machine (Ramadhan 1992a). Eisenstadt *et al.* (1992) are of the opinion that the traditional ITS for programming will not succeed and any CBL systems for novices needs software visualisation and other supports, rather than teaching alone. This type of belief and the advancement of hardware technology made it possible to create several data structure visualisation and memory animation tools (e.g. BALSA (Brown 1987)) and other text-based support tools (e.g. SODA (Hohmann *et al.* 1992)) for different aspect of programming.. The classic system BRIDGE (Bonar *et al.* 1986) may be considered as an intelligent support environment.

BRIDGE provides an environment for learning program analysis, design and coding. It supports the learner to develop relevant code in an evolutionary fashion from a problem given in natural language. Firstly, to solve the problem, higher level goals (textual phrases) are selected from a menu. If the solution is correct, the plans are formed for each goal (using a menu again). Finally, the code segments are selected to facilitate the plans. BRIDGE and PROUST dissect problems into goals and these goals into plans. However, BRIDGE works in top-down fashion (goal to code) whilst PROUST works bottom-up (code to goal). BRIDGE provides a declarative view of the problem (but it is intended to teach a procedural language). It is an interesting feature, since even declarative languages, such as LISP, are usually taught in procedural fashion within CBL systems (e.g. LISPITS).

Computers are now powerful enough to provide complex interactive graphical tools in real time, in order that the learners can visualise the dynamics of the underlying notational machine. Nearly all the recent research in CBL systems for programming includes visual

supporting environment. Meta studies on visual support environment for programming can be seen in Bednarik *et al* (2005) and Hundhausen *et al* (2002).

An remarkable example of a visual supporting environment is JELIOT-3 (Moreno *et al.* 2004). It is more than a program visualisation tool designed for learning and teaching Java. Learners may interact with the system during visualisation, and therefore, it provides an active learning environment. Furthermore, Bednarik *et al.* (2006) are conducting research into incorporating eye tracking for identifying learners' intentions. In future, Jeliot-3 may include a Learner model, and may become a full-fledged collaborative intelligent learning environment (Bednarik *et al.* 2006). The system has been tested and shown to be useful for novice students with difficulties in programming.

Intelligent Learning Environments

Intelligent Environments are similar to ITSs, with an additional supportive environment (or in other words, support environments with significant intelligent support). Typical examples include ITEM/IP (Brusilovsky 1995), and DISCOVER (Ramadhan 1992b), and ELM-PE for LISP (Weber 1993). ELM-ART (Brusilovsky *et al.* 1996) is a web-based intelligent environment based on ELM-PE. There has been very little research undertaken recently into intelligent learning environment for programming. Exceptions include a web based system for learning Lisp programming (Wong *et al.* 2003), and a learning environment for debugging (du Boulay *et al.* 2003).

ITEM/IP is claimed to be an integrated intelligent environment that supports various features, such as knowledge sequencing, task sequencing, editing environment, debugging and investigation etc. DISCOVER provides a visual conceptual environment that enables novices to relate problem solving with both language and machine. Also, it automatically analyses the student's program for logical and semantic errors, and provides intelligent feedback.

Wong *et al.* (2003) discuss an agent based learning environment that facilitates reciprocal tutoring of functional programming on the net. The authors claim that the tutor agent in their system is similar to Lisp Tutor (Anderson *et al.* 1985). They also claim that their

primary programming tool adopts the design of PETAL, a mental-model-based programming environment (Bhuiyan *et al.* 1994).

Du Boulay *et al*. (2003)) describe an interesting attempt to design an intelligent environment for learning debugging skills. The CRUSADE project (Romero *et al.* 2003), which is the origin of the above idea, aims at exploring the role of perspective, modality and individual differences in program comprehension, and designing a visual debugging environment for novice programmers and learners of programming. The proposed tutor is based on the experimental results of two debugging empirical studies (du Boulay *et al.* 2003). The first experiment was performed with a static Software Debugging Environment while the second one employed a dynamic, more interactive Software Debugging Environment. du Boulay *et. al.* conclude,

> "*These studies suggest that good debugging performance is associated with taking advantage of the resources available in the debugging environment. For modern, multi-representational debugging environments, two important resources are the supporting visualisations and the facility to execute the program in steps. A crucial skill in taking advantage of these resources is the ability to decode the information comprised in the supporting visualisations*".

3.3 Formal Methods in Software Engineering

3.3.1 Overview

Formal methods are mathematically-based languages, techniques and tools for specifying and verifying software/hardware systems. The goal of using formal methods is to enable developers to construct software/hardware systems that operate (comparatively) reliably. The demand for higher-quality software is always escalating. Formal methods have already demonstrated enormous success in safety-critical software developments. 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. Some formal methods use a model-based approach for sequential systems (e.g. VDM, Z, Larch (FME 2006)), some use algebraic relationships (e.g. OBJ (FME 2006)), some are designed for object oriented modelling (e.g. Z++, Object-Z, VDM++ (Lano *et al.* 1996)),

and some are useful for concurrent systems (e.g. CSP, CCP (FME 2006)). It is essential to select the right method for the right application. The Z notation is based on Zermelo-Fraenkel set theory and first-order predicate calculus. For example, Figure 3.1 gives a part of formal specification in Z language for Credit card application. The Object-Z specification for the same problem is given in Duke *et al*. (2000). In the late 1970s, the Programming Research Group at the Oxford University Computing Laboratory started the process of developing Z (Bowen 1994). It is one of the most widely used formal method notation in industry for software specification. Moreover, it is also standardised.

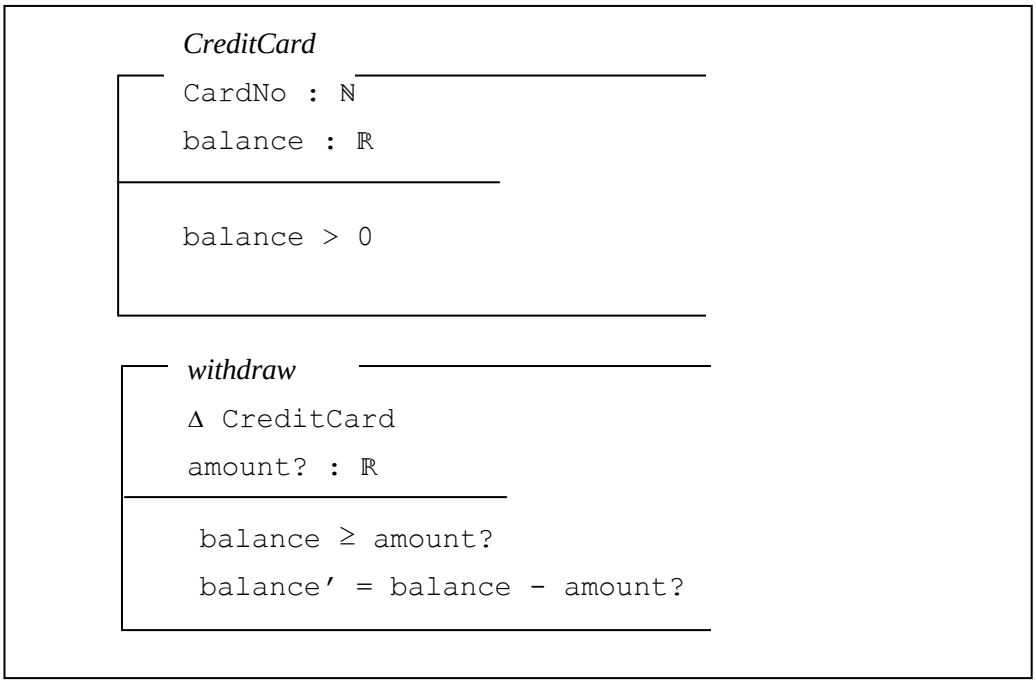


Figure 3.1 Example: Z Specification for Credit Card Application

Although formal methods are used in various tasks, such as software/hardware specification, verification (model checking and theorem proofing) and animation, the key notion of formal methods is clearly formal specification, which is the foundation on which all further development processes are continued (Tremblay 2000). The tangible benefits of using formal specification in software development processes include: (a) design flaws, inconsistencies, ambiguities, and incompleteness can be discovered easily in the early stages; (b) reliable systems may be developed in less time and with less cost; (c) test-suits may be created easily; (d) code may be verified automatically; and (e) code may be generated automatically. An important intangible benefit is gaining a deeper understanding of the system being specified.

3.3.2 Formal Method Tools

Being a mathematical notation, formal methods cannot be applied to large-scale software developments without appropriate tool support (Craigen *et al.* 1993). As the role of formal methods becomes important in all the phases of the software development process, the tools and methods used in this process should ensure seamless transition between different phases. Usually, a formal specification document contains formal expressions with informal annotations, based on (typed) set theory and predicate logic.

There are more than one hundred tools available for various formal methods. About thirty of them were created for the Z notation alone (Steggles *et al.* 1994) Almost all of them provide a syntax directed editing facility in order that a specification document may be created or easily edited in the environment. These tools can provide help with language syntax in batch mode or on-line. Symbols may be inserted by selecting icons or using tags For example, Figure 3.2 gives an interface of a formal method tool Z/EVES (Saaltink 1997). Documents may be viewed in LATEX or some other suitable form, and may be printed in graphical notation (pretty printing). Consistency checking and type checking may be performed in batch-mode. Some tools provide cross-links and expand/contract points for easy browsing. Others include querying facilities so that a user can find out the static semantics of a specification item. Many tools provide libraries and support multi-user projects. Advanced tools support refinement, that is, specification can be refined to executable code in steps. Some tools support verification via theorem proving (Steggles *et al.* 1994). None of these types of tools have any direct pedagogical components attached to them.

Formalizer (Flynn *et al.* 1989), for example, is a syntax-directed editor, browser and type checker for Z notation. It does not have a proof or refinement facility and at the most, from a pedagogical point of view, it gives short error reports (Figure 3.3). Almost all the other tools intended for Z notation are similar to Formalizer. A detailed comparison can be seen in Steggles (1994) or the FME Tools Database (FME 2006). The following tools are notably different from others since they claim to be designed as teaching aids: Zbrowser (Mikusiak *et al.* 1995), VisualiZer (Yap 1999), ZTC/ZANS (Jia 1995b, a) and ZED/ZAL (Morrey *et al.* 1993). A detailed discussion on these tools is included in Section 3.3.

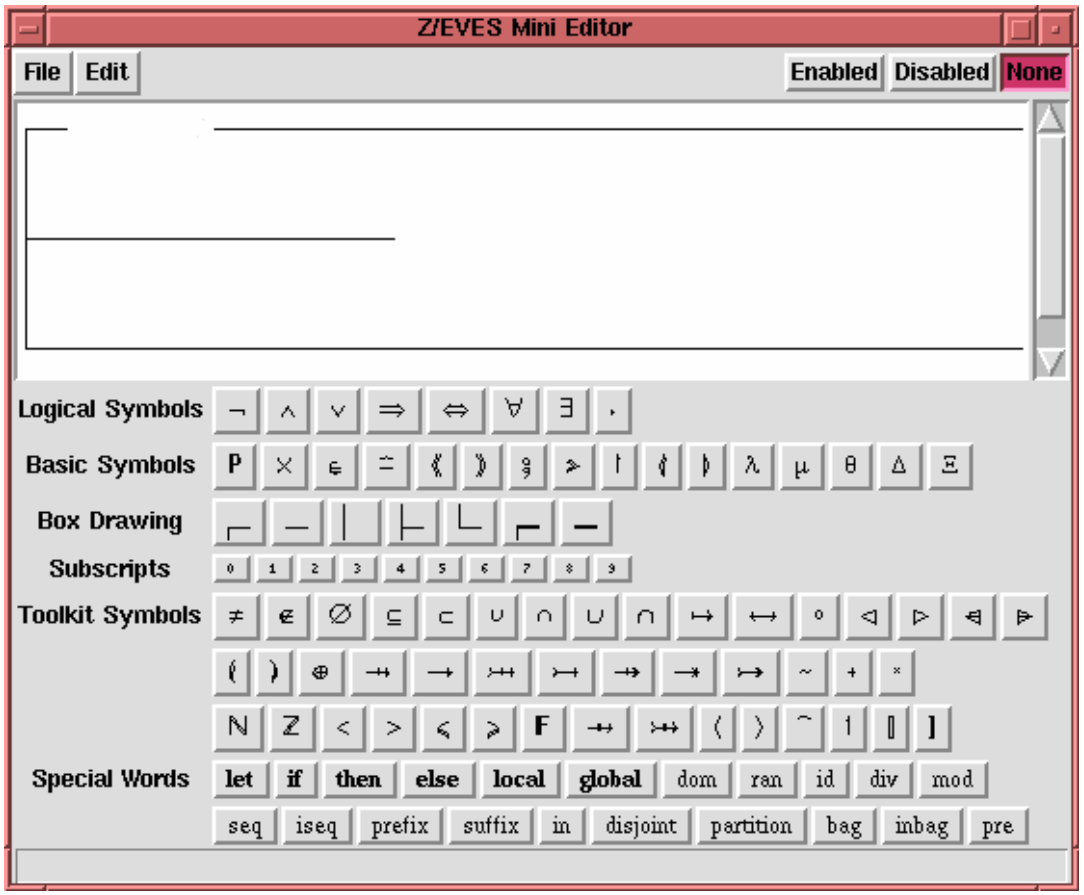


Figure 3.2 Z-EVES environment, a formal method tool

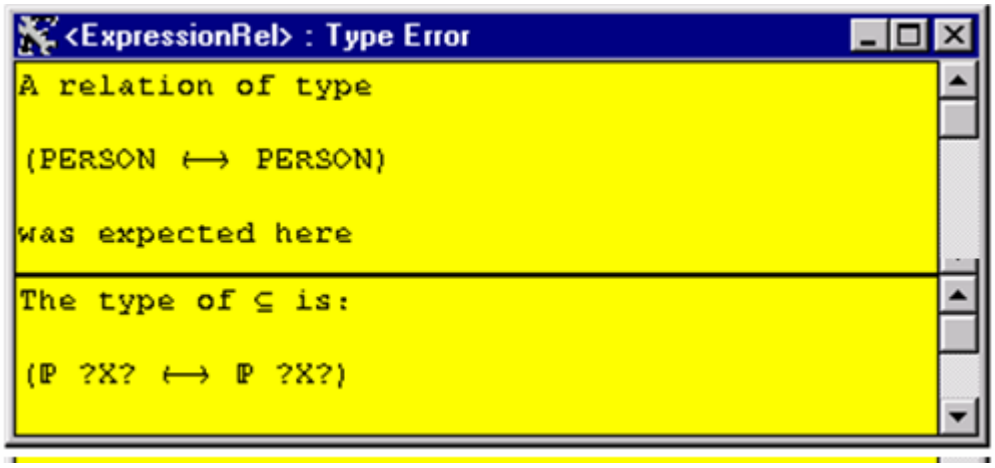


Figure 3.3 Formalizer Feedback

3.3.3 Teaching and Learning Formal Methods

Generally, the subject 'formal methods' is taught in universities in three different disciplines; Mathematics, Computer Science and Software Engineering. Tremblay (2000) maintains that formal methods is not a mathematical subject. Jonathan Bowen, a formal methods expert, suggests, from his experience in teaching formal methods to undergraduate courses, that the specification aspects of the formal method should be emphasised in the initial stage and the analytical aspects may be introduced later, if necessary (Bowen 2000). He also states that using computer based tools gives better results than a traditional paper-pencil approach (*ibid*.). At Massey University, New Zealand, formal methods are taught primarily to Software Engineering students.

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 software practitioners. Jonathan Bowen's experience speaks for the integration of computerised tools in his courses (Bowen 2000). In their edited book, Dean *et al.* (1996) explain their role-play approach to teaching formal methods. Gibson *et al.* describe 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"* (Gibson *et al.* 1998, pg. 1).

The following are some of the reasons stated in the literature relating to students' difficulties in learning formal methods:

- Insufficient mathematical ability
- Complex notation and structure
- Lack of motivation (at the beginning) (Mikusiak *et al.* 1995)
- Inability to abstract details (Duke *et al.* 2000)

The last three difficulties may be alleviated by using appropriate artefacts in a CBL system. More on this issue will be discussed in Chapter 6.

3.4 Teaching Systems for Formal Methods

The software systems that are primarily designed for learning formal specifications fall into two categories. The first category includes all the formal method tools that are claimed to be useful for learning. They are, in general, developed by researchers in the formal specification discipline. The second category includes systems which are principally developed for pedagogical purpose by researchers in the CBL systems discipline. Each category will be considered in turn.

3.4.1 Formal Method Tools for Teaching

The research literature related to the following software tools claims that they could be used for learning formal methods: Zbrowser (Mikusiak *et al.* 1995), ZAL/ZED (Morrey *et al.* 1993), VisualiZer (Yap 1999), and ZTC/ZANS (Jia 1995b, a). Basically, they offer an environment for the users to actively learn a specific topic through trial and error. All of these tools demand significant tutor guidance for novices. Some of them, though primitive, do provide useful hints and feedback. The first three of these tools will be discussed in detail.

Zbrowser (Mikusiak *et al.* 1995) is principally a syntax-directed browser for Z. It lacks data type checking, proof or refinement facilities. Primarily, Zbrowser is designed to overcome the second difficulty mentioned previously – complex notation and structure. The authors claim that it can be used as a teaching aid. There are features which do differentiate Zbrowser from many other tools for Z notation, such as the graphical representation for 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. From a pedagogical point of view, all the above mentioned features make Zbrowser an efficient teaching aid. Zbrowser was evaluated with 40 subjects and the results show (not statistically) that the subjects using Zbrowser performed better than the control group, in both quality and comprehension of Z specifications.

The package ZAL/ZED (Morrey *et al.* 1993) 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. ZAL primarily addresses the third difficulty mentioned previously, whilst ZED addresses the second difficulty. This package, Morrey *e. al.* (*ibid*.) maintain, encourages and facilitates an exploratory (rather than declarative) approach to formal specification, and in turn supports the teaching process, since this allows students to use a 'try and see' approach. Although no formal evaluation is reported, the authors, based on the students' feedback, provide empirical evaluation and assessment of the 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, p. 331). Formal specifications in Z can be transformed to LISP code automatically. The learner can inspect the code and explore the real impact of their specifications. ZAD/ZEL utilizes the transformation technique (formal specification to program) for pedagogy. However, the construction process is largely undertaken by the system. Nevertheless, the 'trial and error' or 'try and see' approach allows the learners to actively examine, understand and develop an insight into the construction process.

VisualiZer (Yap 1999) is a visual environment used to create Z documents. It has significant features to specify data abstractions. With the exception of this, it does not have many interesting features, other than the graphical editing tools. Evaluation is not reported.

3.4.2 CBL Systems for Formal Methods

Despite the number of tools for aiding the development of specifications, surprisingly, there appears to be no reference in the literature for CBL systems for formal methods. MEMO-II (Forcheri *et al.* 1994) and FLUTE (Devedzic *et al.* 2000) are the only existing CBL systems found to be related to this research. As noted by its authors, MEMO-II is intended for learning programming not for formal specification. It is an education oriented programming environment, which allows users to build programs from formal

specifications via interaction with the system. Forcheri *et al.* (1994) 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 *et al.* (1994) claim that it also offers facilities to map this abstract model into effective implementations.

MEMO-II guides novice programmers to build abstract models and experiment with them in different programming paradigms. The underlying formal method used in this system is an algebraic specification language. Firstly, MEMO-II provides a syntax supportive editing environment, within which a specification may be built interactively. Prototypes cannot be created automatically for all the specifications. Therefore, secondly, it automatically checks whether a prototype may be generated for this specification. The types of errors, if any, are explained in detail. Thirdly, it checks whether the specification owns and only owns the intended properties. This proof process is carried out automatically. Fourthly, the specification can be translated into a programming language in different paradigms (functional-LISP, procedural-C and logical-PROLOG). Finally, the resultant code can be executed within the Memo-II environment itself.

MEMO-II, in the same way as a formal method tool, provides a syntax sensitive editing/browsing environment, and validating, proving and animating capabilities with its own compiler. From a pedagogical point of view, it can be used as a learning system within which learners can study by experimenting and using suitable examples. However, learners are left to make their own comparisons between specifications and the resultant generated code. Nothing is mentioned about evaluating MEMO-II. In a similar way to ZAD/ZAL, MEMO-II also uses a transformation technique (from formal specification to computer program) for pedagogy. However, ZAD/ZEL uses it for learning formal specification, whereas MEMO-II uses it for computer programming. In order to learn a computer programming language, MEMO-II requires novices to learn a formal specification language, the syntax of which is equally or more complicated. Usually, formal methods are taught after programming. MEMO-II may also be useful for revising algebraic formal specification.

The other CBL system that has some linkage with this 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 the underlying theoretical aspects of formal methods than its application to software engineering. It is more useful for computer science students than software engineering students or practitioners. The authors compare FLUTE with other ITSs, but no classroom evaluation is reported.

3.5 Summary

Learning formal methods and programming is considered challenging. Due to its high linkage with mathematics and logic, software tools are inevitable in the use of any formal methods. Some formal method tools, such as visual programming environments, provide suitable features for learning some aspects of formal methods. Besides, only one CBL system is designed to teach formal methods. Another system, although it is designed to learn programming, may be used to learn certain aspects of formal methods. For all that, there are still no systems designed to learn any object oriented formal methods.

Being formal notations, learning formal notation and computer programming have many common features. There are hundreds of CBL systems designed for programming. They are grouped under four headings. Firstly, Bug Finders employs a trial-and-error approach in teaching programming - coding and debugging, in contrast to abstraction and refinement. Feedback will be given only after the full program is submitted. Secondly, Intelligent Tutors can monitor each step and provide appropriate assistance at the right time. The system control may vary from model tracing to mentoring. Thirdly, Programming Environments offers an appropriate environment (it includes rich tools such as algorithm animation, data structure visualisation etc.) for the learner to build their own understanding. Fourthly, Intelligent Programming Environments include both features of rich interactive environment and adaptive assistance.

Chapter 4

Learning by Transformation and Transforming UML models to Object-Z

4.1 Introduction

This research investigates whether the problem transformation techniques can be used in CBL systems for certain complex domains in order to reduce the difficulty associated with learning such disciplines. As an exemplar, an object oriented formal notation, Object-Z, and the semi-formal notation UML (Unified Modelling Language) are selected as the subject and support domains, respectively. The transformation process from UML models to Object-Z models is used to design suitable artefacts for active learning. This chapter includes a literature review related to the above topics.

Section 4.2 gives overview on the key research activities that attempt to use problem transformation to build active learning environments. Secondly, Section 4.3 outlines the main research activities that investigates semi-formal model to formal model transformation processes.

In Sections 4.3.1, a brief introduction to the Requirement Specification process is given. Sections 4.3.2 and 4.3.3 outline UML and Object-Z notations respectively. Section 4.3.4 describes, in general, the past research towards formalizing semi-formal models. Section 4.3.5 then discusses the research initiatives related to transforming UML models. Section 4.3.6 covers a very special case where the key issues related to transforming UML models to Object-Z are discussed in detail.

4.2 Learning by Transformation

The significance of "active learning" is being promoted by researchers in education for many decades. As Laurillard (2006) put it,

> "*Whatever their original disciplines, the most eminent writers on learning have emphasized the importance of active learning. The choice of language may vary:*
>
> - *Dewey's inquiry-based education*
> - *Piaget's constructivism*
> - *Vygotsky's social constructivism*
> - *Bruner's discovery learning*
> - *Pask's conversation theory*
> - *Schank's problem-based learning*
> - *Marton's deep learning*
> - *Lave's socio-cultural learning*

but the shared essence is the recognition that learning concerns what the learner is doing, *rather than what the teacher is doing, and the promotion of active learning in a social context should be the focus of our design of the teaching-learning process"* (Laurillard 2006, p. 73).

As mentioned in Chapter 2, Piaget's constructivism emphasizes that the learner needs to construct their own knowledge based on their previous knowledge. Ackermann expresses a more radical approach,

> *"To a constructivist, knowledge is not a mere commodity to be transmitted – emitted at one end, encoded, stored, and reapplied at the other– nor is it information, sitting 'out there' and waiting to be uncovered. Instead, knowledge is (derived from) experience, and actively constructed and re-constructed by subjects in interaction with their worlds" (Ackermann 2007, p. 149).*

In addition to constructivism, Vygotsky (1978) emphasises the impact of society and language in knowledge acquisition. He stresses that learning is possible only if the new knowledge is apprehensible with relevant social support. In Shank's problem-based learning, students were given relevant problems in the subject domain that they need to

try to solve with the aid of mentors and peers. Constructionism, an instructional theory invented by Papert, suggests that in order to construct knowledge learners should engage in some sort of meaningful creational activity. While comparing Piaget's constructivism and Papert's constructionism, Ackemann put it,

> *"Because of his focus on learning through making (on could say learning as design) Papert's "Constructionism" sheds light on* how people's ideas get formed and transformed when expressed through different media, when actualized in particular contexts, when worked out by individual minds. *The emphasis has shifted from general laws of development to individuals' conversation with their own representations, artifacts, or objects-to-think with" (Ackermann 2004).*

She further adds *"Unlike Piaget, Papert thinks that "diving into" situations rather than looking at them from a distance, that connectedness rather than separation, are powerful means of gaining understanding. Becoming one with the phenomenon under study, in other words, is a key to learning." (ibid, p. 20).*

An instructional strategy called Problem Transformation is a special kind of constructionist approach which encourages students to actively involve in creating relevant new artefacts by systematically transforming some existing artefacts. The next section describes this approach in detail.

4.2.1 Problem Transformation

Traditionally, instructors use analogies and metaphors for teaching complex concepts. Problem transformation is a particular kind of constructional approach wherein the learners are encouraged to create some material based on existing materials, or to systematically transform some artefacts from one representation (source) to another one (destination), or to transfer a given problem into an alternative form (Kemp *et al.* 2001). The source and the destination may have the same, or different, levels of complexity. In any case, the transformation process should involve a significant constructive task; it should be challenging and motivational and within the learner's cognitive ability. If the transformation process involves complex cognitive skills, the scaffolding technique may be used. The topic to be learned may be the source discipline, or destination discipline, or the transformation-process itself. Similar to the other constructionist learning approaches,

the proponents claim (Kemp *et al.* 2001), Problem Transformation will also engage learners actively in the relevant knowledge building process. Moreover, as the transformation involves some pre-existing model, learners are brought into the relevant context fairly quickly. This is an additional advantage based on constructivist learning theory.

There is very little theoretical knowledge in this specific stance, and also there are only a few empirical studies have been reported in the literature. Some examples for different types of problem transformation approaches (particularly used in learning systems for programming and formal methods) will be given next.

Some Example Systems

In ZAD/ZEL (Morrey *et al.* 1993), transformation of specifications from a formal notation to computer program is utilized. ZAD/ZAL aims at teaching the source discipline- formal methods. Though scaffolding is not used explicitly, the transformation process is partially automated and the animation process is hidden from the learner. Also, in MEMO-II (Forcheri *et al.* 1994) transformation from a formal notation to a computer program is used for instruction. Similar to ZAD/ZAL, the transformation process is partially automated. However, MEMO-II aims at teaching the destination discipline-computer programming.

Sun *et al.* (2001) describe a system that converts Object-Z specification to UML models, which they claim could be used for teaching semantics of both Z and Object-Z. This system aims at teaching both source and destination disciplines. The system transforms a model from a complex domain to a less complex domain. A software development process may follow the following order: informal specification (text) → semi-formal specification (UML) → formal- specification (Object-Z) → code (Java). Usually, more information will be added to the model during the transition. However, during backward transformation some potential information may be lost.

Kemp *et al*. (2001) describe various examples of this approach. A concrete recent example is Leopard Tutor (Kemp *et al.* 2003). It is a visual supportive environment that uses transformation processes from computer programs to UML models in order to learn

programming concepts. Basically, learners are encouraged to create class diagrams from computer programs to learn the fundamentals of object-oriented computer programming. Similar to ZAD/ZAL, Leopard Tutor is also designed to learn the source discipline. Leopard Tutor, like Sun *et al.*'s system (2001), uses backward transformation. However, unlike ZAD/ZAL – where the system does much of the transformation process Leopard enforces active learning by encouraging the learners to perform the whole transformation process (although some support is given). Both, understanding a solution of a problem and designing a solution to a problem involves different but related cognitive skills. Leopard Tutor, the authors claim, is designed to bridge such gaps in object oriented programming. In the 2003 version, learners are asked to create relevant class diagrams from the given Java code (and comments). Comments are used to build the semantic meaning of the program (the authors call this a task level class diagram).

In this study, students will be engaged in transforming UML models to Object-Z specifications. UML is a graphical semi-formal notation, and Object-Z specification includes complex mathematical notation. Transforming UML to Object-Z is a significantly challenging process. The next section outlines the current state of the research related to this transformation process.

4.3 Transforming UML models to Object-Z Specifications

UML and Object-Z notations are used for specifying requirements, primarily in the OO software development process. Object-Z is a formal language, whereas UML is a graphical semi-formal notation. UML is easy to learn and use, but ambiguous (France *et al.* 2000). Object-Z, though challenging to learn, gives precise specifications (Amalio *et al.* 2003). Automatically transforming UML specifications to Object-Z specifications is obviously useful. However, it is also challenging and has been one of the major research areas related to Formal Methods discipline. In this research, the proposed instructional strategy uses problem transformation to alleviate the difficulty in learning Object-Z notation. Using this technique, students are encouraged to be actively involved in UML to Object-Z transformation process for selected case studies. Obviously, for novices, transforming UML models would be easier than creating Object-Z specifications from scratch. Therefore, in early stages of the learning process UML models can serve as scaffolds; and as the student progress, they would be able to directly prepare Object-Z

specifications. In order to incorporate this mechanism in the CBL system, a simple step-by-step methodology for UML to Object-Z transformation process is essential. This research proposes such a methodology for transforming simple UML class diagrams to Object-Z (Chapter 7).

4.3.1 Requirement Specification - Overview

The requirement specification stage is one of the major phases in software development. The specification document plays a major role in the software development process. Software contracts are based and signed on this document. Moreover, the remainder of the development processes will also be based on this document. Therefore, it should be clear, concise and complete (Sommerville 2001). The specification may be just a collection of textual descriptions: in this case, ambiguities are inevitable, due to the nature of informal natural languages. At the other extreme, the requirements may be defined using formal notation, such as Z or Object-Z. If this is so, then it will have unique meaning, and the structure and the behaviour of the expected end product will be clearly specified. However, this accuracy cannot be obtained without a cost. Formal notations are difficult to learn and understand (Fowler 1998). There is, nonetheless a middle ground called semi-formal notations such as Entity-Relationship diagrams, Data Flow Diagrams, Structure Charts, and UML used for defining specification. Being graphical notations, they are relatively easy to learn and use. They all have different degrees of formalism.

4.3.2 UML Overview

UML is just a collection of complex graphical notations, albeit a powerful one, used in the OO software development process (Booch *et al.* 1999). It includes a number of different diagrams for use in different phases of the software development process. It can also be used to model hardware functions, business processes, etc. UML is an outcome of combining a collection of best practices within the software development process (Fowler 1998) and is a de-facto standard for OO software development notation. The main artefacts are defined by UML notations themselves. It can be extended through the use of stereotypes. The use case diagram, class diagram and interaction-sequence diagram are all important diagrams in the UML notation. Their function will be briefly described in the following section with the discussion being mainly based on Booch *et al.* (1999), Bennett *et al.* (Bennett 2000), and Fowler (1998).

Use Case Model

A use case model is primarily used to capture requirements. It specifies the requirements from the users' point of view and from the outside inwards. This model includes use case diagrams, use case specifications, and activity diagrams (optional). The use case diagram clearly shows the boundary of the intended system, the potential users of the system and their main requirements. In addition to the diagrams, for each use case, a use case specification will be used to describe the user-system interaction in detail. The requirements may be grouped and ranked.

A case study is used in this chapter to illustrate the key artefacts of both UML and Object-Z notations. This class study is a modified version of the problem given in Duke *et al* (2000, chapter 3), and, therefore, the reader may examine the varied solutions in slightly different situations.

Case Study

A building has many rooms. Each room has at least one key (a magnetized card). A room may be in a locked or unlocked state at any time. Opening a given room is considered the main requirement of this system. Figure 4.1 gives a portion of use case model for this case study.

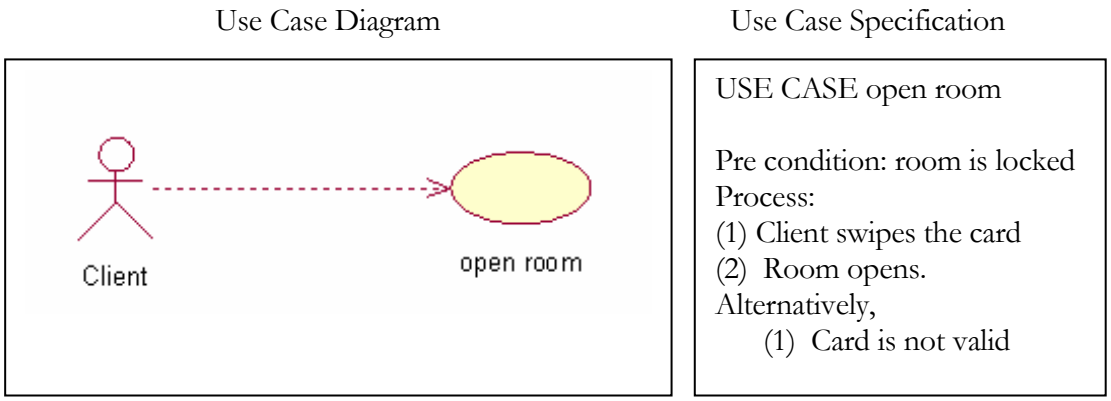


Figure 4.1 Use Case Model**:** Key-Room Case Study

Class Diagram

The class diagram gives the static structure of the required system. It includes the potential classes and their relationships. The classes and their inter-relationships (associations, inheritance, and compositions), at the analysis level, resemble real world

entities and their interactions in the current context. The classes are templates for objects. A class includes attributes and methods. All the features of a class are annotated by their visibility: public, private or protected. The relationships are annotated by the multiplicity constraints. Some classes may be generalizations of more specific classes. Some classes may be a composition of other classes. An OO system, in the static view, is a collection of classes. Whilst executing, it is a collection of objects that interact with each other to perform specific tasks. Figure 4.2 gives a portion of the analysis level class diagram for the case study.

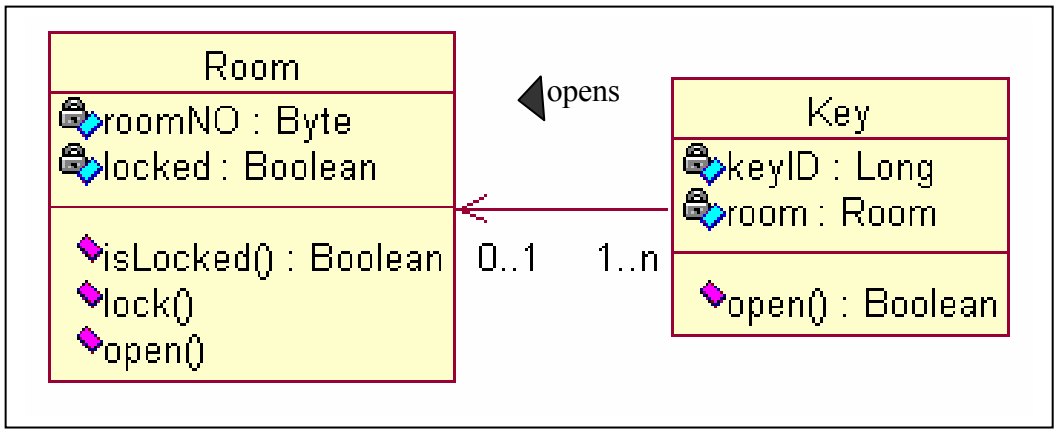


Figure 4.2 Class Diagram: Key-Room Case Study

In specifications, operations are usually declared as contracts. Figure 4.3 gives the specifications of the following operations: 'open' for Key class, and 'open' and 'isLocked' of Room class. Note that the association is already designed to include the associated Room object as an attribute in the class Key (in Figure 4.2, it is 'room').

Sequence Diagram

Objects interact with each other to realize a requirement. A sequence diagram specifies how and in which order the objects need to interact, in order to perform certain tasks. For each use case requirement, a number of sequence diagrams may be drawn to specify all the possible realizations for different alternatives. Both class diagrams and sequence diagrams should be consistent. Figure 4.4 shows the sequence diagram for 'open room' use case realization (main case).

Class : Key
Module : open
Parameter: room: Room
Input: none
Output: success: Boolean
Precondition: none
Postcondition: if (room = self.room)
room.open()
success ← true
else
success ← false
End Open

Class : Room
Module : open
Parameter: none
Input/Output : none
Precondition: self.locked=true
Postcondition:
self.locked=false
End Open

Class : Room
Module : isLocked
Parameter: none
Output: yesNo: Boolean
Precondition:
Postcondition:
yesNo ← self.locked
End isLocked

Figure 4.3 Operation Specifications: Key-Room Case Study

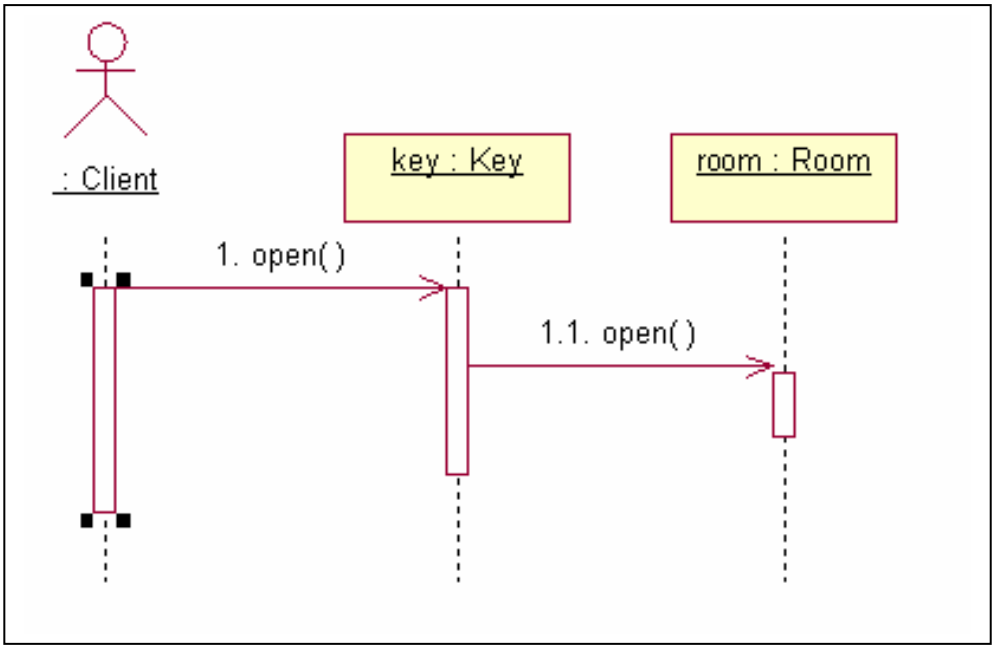


Figure 4.4 Sequence Diagram: Open-Room Use Case

4.3.3 Object-Z Overview

Object-Z is an OO formal notation based on Z (Duke *et al.* 2000). Z is a state-based formal notation (Barden *et al.* 1994). The class schema in Object-Z is a named entity,

which encapsulates the state schema and a collection of operation schemas. Figure 4.5 outlines the structure of the class schema. Schemas in Object-Z, similar to Z, have two parts: the declaration part and the predicate part. The declaration part primarily lists the relevant identifiers with types, and the predicate part contains constraints pertinent to the schema, in first-order predicate logic. If there are no such constraints, the predicate part may be omitted.

Object orientation views a software system as a collection of interacting objects; therefore, the specification of a system will contain a collection of class schemas. Each class can be easily understood in isolation. There are some composition operations (e.g. the parallel operator) available in Object-Z to cater for message passing between objects. Special constructs are available to specify inheritance and polymorphism. The notion of Object containment is used to specify object compositions. The class union feature in Object-Z, which is not available in UML, facilitates rich class associations. Some of the important constructs are explained in the following paragraphs. For more information, refer to Duke *et al* (2000).

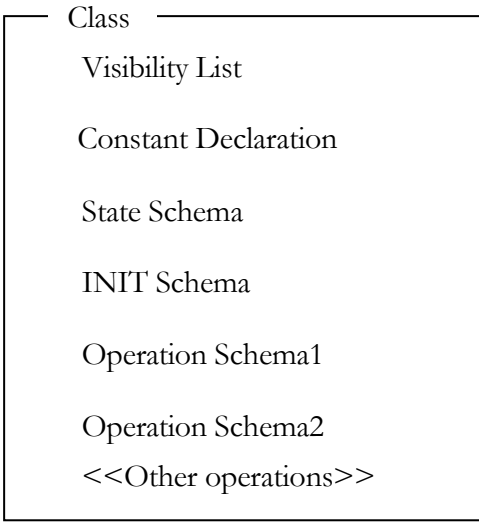


Figure 4.5 Structure of a Class Schema

Visibility List: The Visibility List includes all the public features of a class. If a schema has no explicit visibility list, all features are visible to the environment.

Constant Definition: In a particular instance of a class, a constant cannot be changed. This is not equivalent to a class variable, which is not explicitly available in Object-Z.

State Schema: This is an un-named schema where attributes are defined. It also includes all the invariants of the class. The relationships between attributes, if any, are also described (using the Delta operator). The initial values for the attributes cannot be included here.

Initial Schema: The initial conditions of a class are defined here (e.g. the initial values for the attributes can be defined). The initial conditions should satisfy the class invariant. The name of the schema is always 'INIT'. It is important to note that it is not an operation that initializes an object. Therefore, if necessary, a separate operation schema needs to be included for initializing an object.

Operation Schemas: Usually, there will be more than one operation schema in an Object-Z class schema in order to specify the operations of a class. Figures 4.6 and 4.7 give the class schemas for the case study. Note that, some obvious structures (for example, INIT in Key class) are not included in the schemas, in order to avoid complexity.

The Room class (Figure 4.6) is independent of the Key class, but the Key class (Figure 4.7) depends on the Room class. The Key class contains a Room object, which maintains the link from a key to its room. Conversely, the link from a room to its key is lost (this is unnecessary, due to the navigational direction – see Figure 4.2). An operation of the Key class utilizes (or depends on) an operation of the Room class (see the sequence diagram Figure 4.4).

A separate class schema may be included to manipulate (add/remove or update) the link between Room and Key objects. A mandatory constraint can be included as an invariant (e.g. for every room, there exists at least one key). This issue will be discussed further in Section 4.7.1.

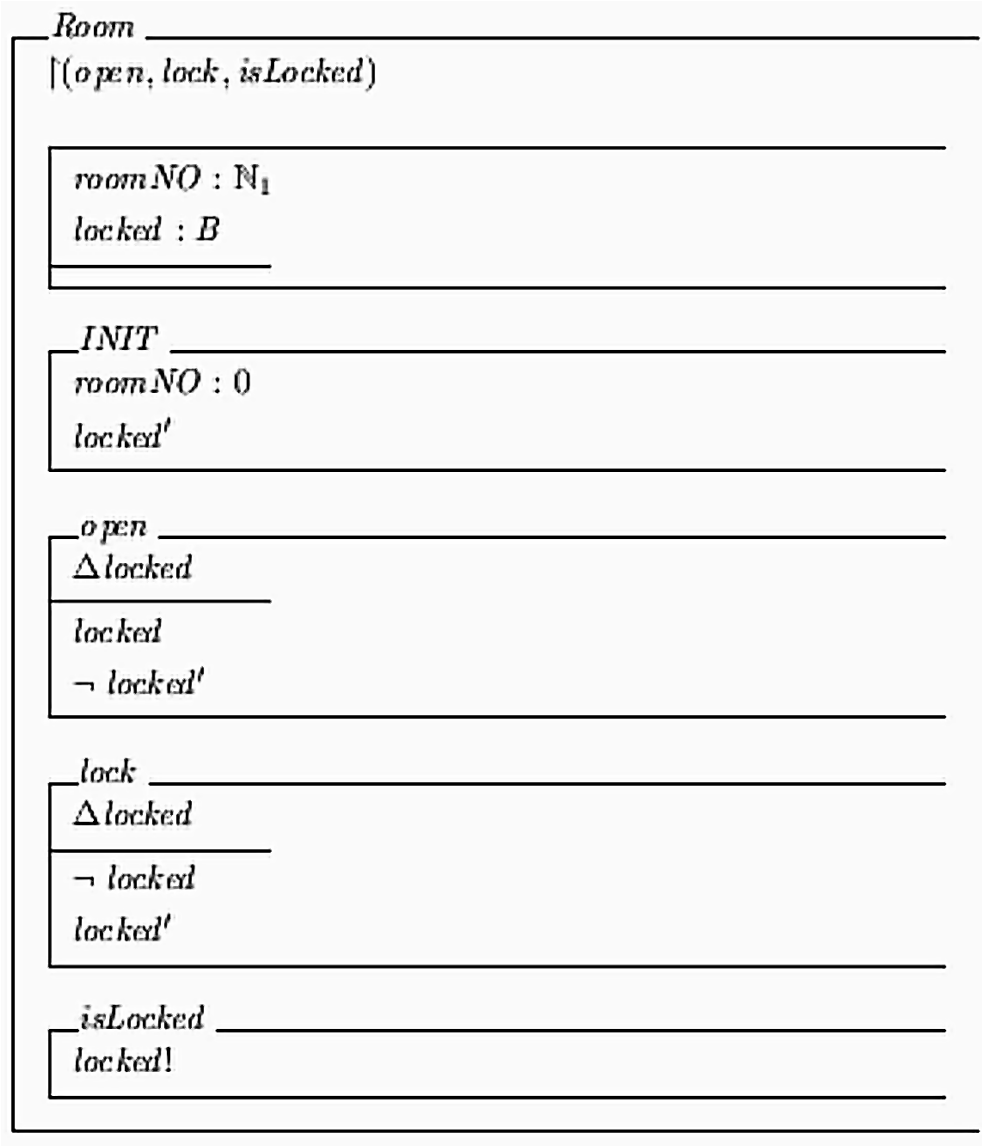


Figure 4.6 Class Schema for Room

4.3.4 Transforming Semi-formal models to Formal models – An Overview

The ultimate purpose of a software engineering process is to reduce the difference between what is required and what is to be delivered. An elegant specification model significantly helps to serve this purpose. Semi-formal graphical languages such as UML have been developed, since they are less ambiguous than informal natural languages, and they are relatively easy for different stake-holders to comprehend. One of the drawbacks of semi-formal models is their dependence on natural languages to describe their semantics, and therefore the specifications may become ambiguous. Nevertheless, semi-formal models are widely used in the software development processes, whereas, formal

models, though concise and unambiguous, are infrequently used in real development tasks. Formal specification is highly recommended for mission critical systems since a clear specification is essential to build the correct system (Dean *et al.* 1996).

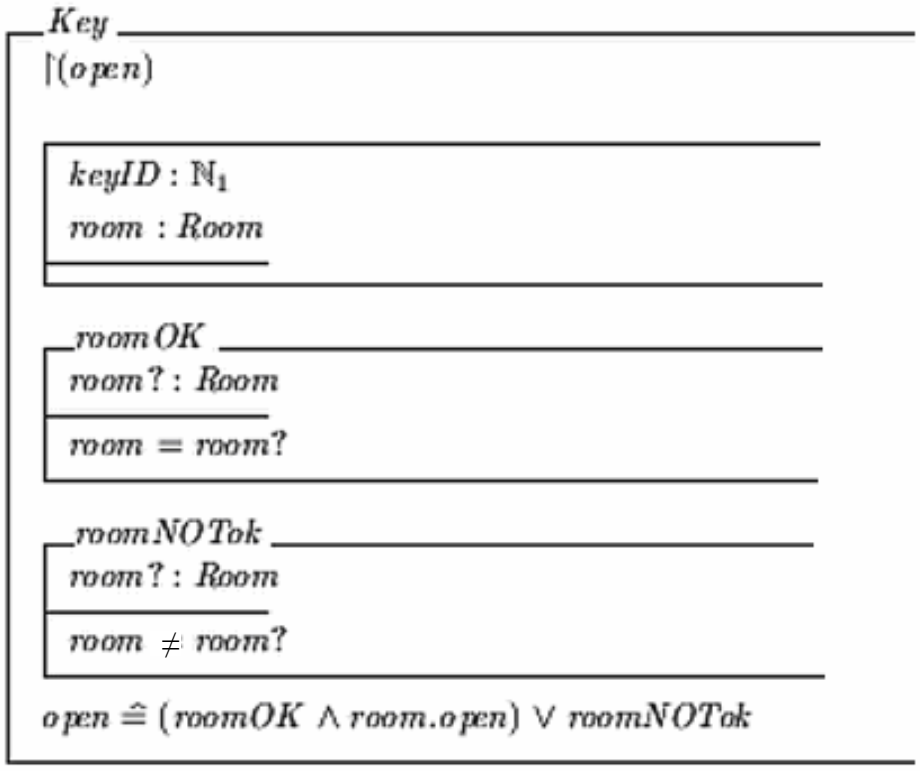


Figure 4.7 Class Schema for Key

Compared to semi-formal models, formal models are difficult to create. Students and software practitioners often find it challenging to learn formal methods. The main difficulty faced by the students, reported by Duke *et al.* (2000) , is not the mathematical notation but the students' lack of ability to relate the notations to important abstractions. A tool that automatically translates a significant portion of semi-formal model specification into a formal representation would be very useful for software practitioners.

A number of researchers have been attempting to address the issue of strengthening semi-formal models by formal notations. The next section will discuss some of the previous research in this direction. Beginning with more general cases, the following discussion will lead to a particular case, where Object-Z is used to formalize the UML notation.

Formalizing Semi-formal models

Semi-formal methods, such as E-R Diagrams, are graphical notations and their semantics are not defined mathematically. David Harel *et al.* state, *"Many methodologies fail to rigorously define the semantics of the languages. Without a rigorous semantic definition, precise model behaviour over time is not well defined and full executability and automatic code synthesis is impossible"* (Harel *et al.* 1997, p.31). Even the international standard notation UML lacks formalism (Harel *et al.* 2004).

There are volumes of literature reporting various methods to bridge the gap between formal and semi-formal models (Bruel *et al.* 1998). All these attempts are generally called 'formalizing' and can be categorized into three groups (France *et al.* 2000).

- Formally defining semi-formal models using formal languages.
- Integrating formal models with semi-formal models
- Transforming semi-formal models to formal models.

In the first approach, researchers try to define the semantics of those models (distinguishing semantics from its syntax whilst considering the usage) using formal languages. Formalizing these semi-formal models enables researchers to further analyse those models (e.g. translating to code). An example of this approach includes (Polack 1992), on which he discusses formalizing E-R Diagrams using Z notation. The special cases, where UML or Object-Z or both are involved in formalizing (or integrating or transforming), will be discussed in detail later.

The second approach, integrating semi-formal models and formal notation, tries to fill or annotate any missing information in the semi-formal models, by appropriate constructs using formal languages. For example, the semi-formal models in SSADM (Weaver 1993) are integrated with Z to form a methodology known as SAZ (Polack 2000). However, some experts in this discipline believe that 'true integration' is not possible between formal and semi-formal methods. In an overview of a panel session on *integrating formal and informal specification techniques*, Bruel *et al.* suggest[3]:

[3] Note that the term 'informal' is used in this quotation to denote all semi-formal models (however, in this dissertation, the term 'informal' denotes textual descriptions).

> *"It might be pointed out that true integration is only between formal methods or between informal methods but not between formal and informal methods. This last type of integration can be viewed either as a translation from graphical to formal or as a way of obtaining graphical models from formal models"* (Bruel *et al.* 1998, p.1).

In the third approach, rigorous rules are proposed to automatically transform semi-formal models to formal models. The rules are usually based on the formal definitions of both models. As stated before, formalization of semi-formal models are essential before defining transformation rules (Harel *et al.* 2004). For example, Dick (1991) describes the process of transforming Data Flow Diagrams to Z. Barden *et al.* (1994) used E-R models as a basis for constructing Z specifications for teaching purposes. Nevertheless, some experts believe that automatic translation is not possible without human intervention. The panel-overview paper mentioned earlier states *"Complete automation of the informal to formal specification is unlikely, given that informal specification are inherently incomplete"* (Bruel *et al.* 1998, p 3).

Hereafter, unless otherwise stated, the term 'formalizing' will be used loosely and it will refer to all three approaches: formalizing, integrating and transformation.

The notion of object orientation adds another level of complexity to formalizing semi-formal notation by a formal language. Depending on the paradigms of semi-formal and formal languages involved, four types of combinations are possible:

- Both notations are of the non-OO paradigm (e.g. E-R Diagram and Z)
- The formal notation is of OO paradigm, but the other is of the non-OO paradigm (E-R Diagram and Object-Z)
- Semi-formal notation is of the OO paradigm, but the other is of the non-OO paradigm (UML and Z)
- Both notations are of the OO paradigm (e.g. UML and Object-Z)

Object orientation is a relatively new technology. The non-OO semi-formal notations and the non-OO formal notations have been in use for many decades. Some OO semi-formal

notations, such as UML, were introduced later; and gradually, the non-OO semi formal notations became insignificant. Eventually, UML became the ISO standard, but there is no standard OO formal language. Note that, in the first and fourth types of combinations, both notations are of the same paradigm. Some classical research deals with the formalizing problems involving the first type of combinations, for example, SSADM by Z (Polack 2000). A significant amount of contemporary research addresses the formalizing problems involving the third and fourth types of combinations. A particular case related to the fourth type, transforming UML to Object-Z, will be discussed in Section 4.6. Research attempts involving the second type, formalizing a non-OO semiformal notation by an OO formal notation, are very unlikely.

For the problems related to the third type, formalizing OO semi-formal notations by non-OO formal languages, it is essential to formally define the OO concepts first, using the corresponding non-OO formal language. Formally defining OO concepts, by using the original version of non-OO formal notations, is very difficult. Therefore, the non-OO formal language should be modified or extended, in order to incorporate OO concepts. Considering *Z* as an example, this issue will be discussed in detail in the next section.

Object Orientation in non-OO formal methods (especially Z)

The focus on object orientation in software engineering has radically changed, not only the structure of the software, but also the practices involved in the development discipline. None of the existing formal methods is capable of completely accommodating OO features. For example, some of the OO features in UML models cannot be represented in Z. Since it is a state-oriented formal specification language, it can accommodate the Abstract Data Type (ADT) concept, but not the class concept. ADTs can roughly represent classes. However, there are some concrete differences between both representations. Cook *et al.* (1991) consider ADTs and classes as orthogonal approaches for data abstraction and there are many differences. The following are the key differences between an ADT and a class, which make it difficult to represent OO concepts in Z using its original form:

- No state for an ADT but an object has state
- Class level features are available, but there are no features available in ADTs.

- Theoretically, the data structure of an ADT is totally hidden. In class, the access of an attribute is controlled by its visibility option (e.g. public or private).
- Inheritance is not possible between ADTs.
- Polymorphism is not possible in ADT (type is fixed for a variable).
- There is no ADT identity, but an object has identity. In general, value semantics is used for ADTs and reference semantics for objects.
- ADTs are usually initialized. Objects need to be constructed and destructed.
- No 'Self' construct in ADTs. An object may refer to itself using 'Self'.

The edited book by Goldsack *et al.* (1996) provides a general view of formalizing object technology. There are two ways, described in the literature for a non-OO formal method to incorporate OO concepts (Stepney *et al.* 1992):

- Modifying a formal language.
- Extending a formal language.

In the first approach, various conventions and styles are introduced to a non-OO formal method, in order to enable it to specify OO features. Hall (1990), Hammond (1994), Dupuy *et al.* (1998), France *et al.* (2000) and Utting *et al.* (2002) all recommend different approaches to specifying OO features using Z. Stepney *et al.* (1992) discuss various such research attempts in detail. In the second approach, new artefacts are designed to include OO features. For example, in the context of Z notation, an extended version, with significantly different semantics from the original, has been invented to incorporate OO features. Notable outcomes of this approach are Object-Z (Duke *et al.* 1991) and Z++ (Lano 1991). The resultant language may become a stand-alone OO formal language (e.g. Object-Z). To the knowledge of the author, there is no OO formal method created from scratch. The next section first outlines the nature of formalism in UML, and later discusses some past research related to formalizing UML in general.

4.3.5 Formalizing UML models

Formalism in UML

UMLv2.0 (OMG 2006) has four parts; Superstructure, Infrastructure, Object Constraint Language, and Diagram Interchange. It includes thirteen types of diagrams. OMG is also

promoting a middleware independent Model Driven Architecture (OMG 2006) founded on UML.

UML is defined by using its own constructs (meta-model). The Object Constraint Language, the formal notation included in UML to supplement formalism, is more implementation oriented and depends on Class diagrams (Jackson *et al.* 1999). The meta-model of UML is described in a combination of formal, semi-formal and informal languages (OMG 2006). The abstract syntax of UML constructs are described using the UML Class diagram and supporting natural language description, and the static semantics is given by well-formedness rules using Object Constraint Language and natural language. The dynamic semantics of UML is, however, expressed mostly in natural language (*ibid.*). Ambiguity is inevitable in UML models, and therefore it is not a fully formal language. However, compared to other semi-formal models, UML is well-defined. Therefore, transforming UML to formal models is considered challenging but nonetheless a reachable goal in research circles.

Formalizing UML models

Being an industry and international standard notation, there have been several attempts reported for formalizing UML using various formal methods. A group of researchers called the pUML group (pUML 2006) have been engaged in this research for a long time and they have attained considerable success. Overgaard *et al.* (2000) also report a great deal of research in this direction. The eXecutable-UML group (Mellor *et al.* 2002) goes beyond formalizing and claims that the model described in the formal action semantics could be executable and testable: and further, the executable UML model compiler can translate it directly into code.

Almost all the key formal languages are used to formalize UML. There have been many research attempts, mainly from the pUML group, describing different techniques to transform UML models (static structure and dynamic semantics) to *Z* specifications (Breu *et al.* 1997; Dupuy *et al.* 1998; France *et al.* 2000). In related research, Hammond (1994) explain how OOA (Shlaer *et al.* 1988) models may be converted to *Z*, and Hall (1994) and Utting (2002) explain how OO concepts may be converted to *Z*. Virtually all the approaches for formalizing OO semi-formal models such as UML with *Z* are rather

complex, compared to formalizing those models using Object-Z. Amalio *et al.* (2003) include a table in their survey, in order to compare the number of schemas necessary to formalize a OO feature in class diagrams in both notations. It shows that the Z notation requires a significantly higher number of schemas, compared to Object-Z. The next section will describe in detail a special case: transforming UML models to Object-Z specifications.

4.3.6 Transforming UML models to Object-Z

Transforming UML models to Object-Z is considered relatively easy, since there are important similarities in the semantics between the two notations. Both consider Class as a template that describes a set of similar objects, and allow the use of a Class as a type. Both notations use reference semantics (unlike Z notation – which use object/value semantics). Kim *et al.* (2002) have carried out extensive research on formalizing and transforming UML models using Object-Z notation. Other significant contributions to the research in this specific area are: integrating UP with Object-Z (Araujo 1996); formalizing UML with Object-Z (Miao *et al.* 2002); integrating Object Modelling Technique to Object-Z (Dupuy *et al.* 2000a); and defining UML constructs using Object-Z (Sun *et al.* 2001). As stated previously, Dong's group (Sun *et al.* 2001) uses temporal logic for sequencing, but Miao *et al.* (*ibid.*) use a variable 'order' to hold time related information (for sequence diagrams).

Kim *et al.* (1999; 2000a; 2000b; 2002) have separately published many papers on formalizing key UML constructs, including use case diagrams, state transition diagrams, and class diagrams, and for inconsistency checking. Kim *et al.* (2003) explain their approach for transforming class diagrams in three steps. Firstly, they formalize the syntax (and semantics) of UML constructs using Object-Z notation. Secondly, they formalize Object-Z syntax (and semantics) using both class diagrams and Object-Z itself. Thirdly, they define a mapping between these two constructs. Transforming static structure of UML constructs (particularly class diagrams) will be discussed in detail in the next section.

In addition to the static structure of UML, some research attempts to formalize the dynamic semantics of different UML constructs in Object-Z; for example, Miao (2002)

for Sequence Diagrams, Dupuy (1998) for Object Management Technology and Kim *et al.* (2002) for State Diagrams. Kim *et al.* (*ibid*.) treat the dynamic semantics of UML (particularly for state diagrams) in two parts; denotational semantics and operational semantics. For the denotational semantics, they include relevant invariants to Object-Z state schema. For the operational semantics, they introduce 'semantic variables' as additional attributes and define the operational semantics using Object-Z operations. Kim *et al.* (2000a) claim that encapsulating the entire definition of an UML construct into a single construct (Object-Z class schema) makes their approach unique amongst others within that type. Nevertheless, the main stream researchers (Mellor *et al.* 2002; Kim *et al.* 2005) are now moving towards discussing the formal aspects of OMG's Model Driven Architecture (OMG 2006).

Transforming UML Class Diagrams

Being the major construct in UML, the research on transforming the class diagram has been given much attention in research circles. There are various approaches to transforming key constituents of the UML class diagram to Object-Z. Amalio *et al.* (2003), in their survey, discuss a number of different approaches reported in main stream formal methods literature for formalizing the UML class diagram to Object-Z (and Z). Amalio *et al.* generally compare Kim *et al.*'s (1999) approach with Araujo's (1996), and to some extent, Dupuy's (2000b). In addition to these approaches,, two others (Duke *et al.* 2000; Miao *et al.* 2002) are also included in the following discussion. In many ways, the following discussion supplements Amalio *et al.*'s survey. The Room-Key case study will again be used as an illustration.

Identity

UML uses reference semantics. The early versions of Object-Z also used value semantics. In value semantics, based on Hall's approach (1994), the notion of 'self' has to be defined explicitly. For example, the Key object in value semantics is given in Figure 4.8.

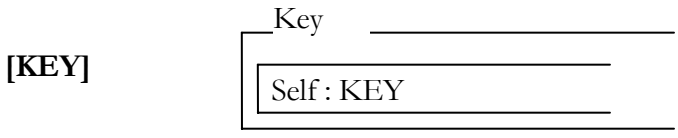


Figure 4.8 Value Semantics and Identity of an Object

Object-Z and UML both use reference semantics, and therefore the semantics for object identity is the same for both. An Object could refer to itself using a built-in construct called 'self' in Object-Z notation. Therefore, the object identities need not be defined as separate types. Moreover, an Object schema is considered as a template and also as a type. In this context, Hall's (1994) extension/intension differentiation is not applicable.

Visibility List

The UML class diagram allows 3 types of accessibility: public, private or protected. There is no such protected access in Object-Z. Whilst transforming the UML classes to Object-Z schemas, none of the above researchers explained how they would treat 'protected' variables. Araujo (1996) notes that the visibility list in Object-Z does not differentiate attributes and methods. In addition to this, operation overloading is not possible in Object-Z, due to the format of the visibility list.

Class Variables

In UML, class features may be included in a Class, and they are differentiated by just underlining. The values of class attributes are common to all the objects of that Class. There is no equivalent construct in Object-Z. None of the research mentioned above touches on this issue.

Construction/Destruction

UML classes may explicitly include constructors and destructors. Existential constraints in an object's link may affect its creation or destruction. Object-Z assumes that the objects are always available in the environment. There is nothing to suggest in the literature as to how this conflict will be resolved, during the transformation of Classes to Object-Z

Association

Class diagrams (and E-R Diagrams) include class relationships, such as association. There is no way to show association relationships in Object-Z and therefore, it should be resolved in some way, in order to represent them in Object-Z schemas. Even in program code, there is no way to explicitly include association. Basically, there are three approaches discussed in the literature: local, recursive and central. The first and last approaches are similar to the approaches proposed in the previous research towards transforming UML to Z notation.

In the first approach, an object identity (or a collection of object identities – depending on the multiplicity constraints) of the participating Classes are included as attributes into the Classes of the opposite sides of the association link. Usually, role names will be used to name those attributes. Depending on the navigational direction, if known, one side of the Class may not need to keep the link, and therefore it may not include the relevant identities as attributes. This approach is related to designing a database from E-R diagrams, where the corresponding primary keys (instead of identities) are included in the relevant tables. Duke *et al.*'s (2000, p. 42.) 'local view' is an good example of this approach. Duke *et al.* include a class schema called 'KeySystem' which consists of a mixture of properties of the relevant control environment (in operation schemas) and the corresponding association (in state schema). It can be noted that the 'KeySystem' is a class schema but it will only have one instance. Figure 4.9 illustrates the first approach.

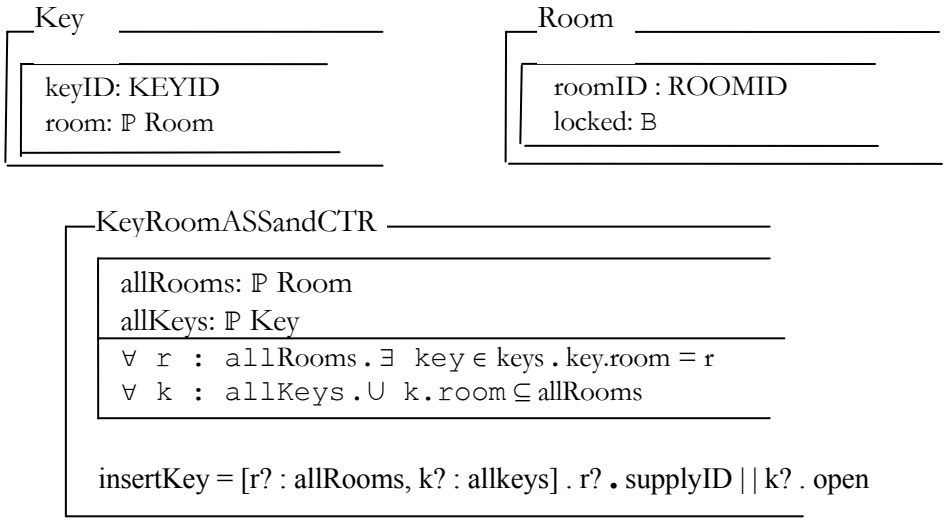


Figure 4.9 Formalizing Association: Local

The features related to control environment (here the operation 'insertKey') may be separated. The operation 'supplyID' will just return the identity of an object 'self' - refer to Duke *et al.* (2000). The parallel operators are used to pass parameters. If the navigation is on both sides, an attribute called keys may be included in the Room class. The 'type' of 'keys' will be a collection of identities of Key objects, which represent the collection of all the keys that open a particular room.

In this second approach, appropriate invariants are recursively included in both classes in order to keep the consistency (Kim *et al.* 1999). Figure 4.10 illustrates the second

approach, as applied to the case study. In a similar way to the first approach, the association links between objects are not explicit and therefore, in this approach, manipulating the association links, whilst keeping the consistency, will be difficult.

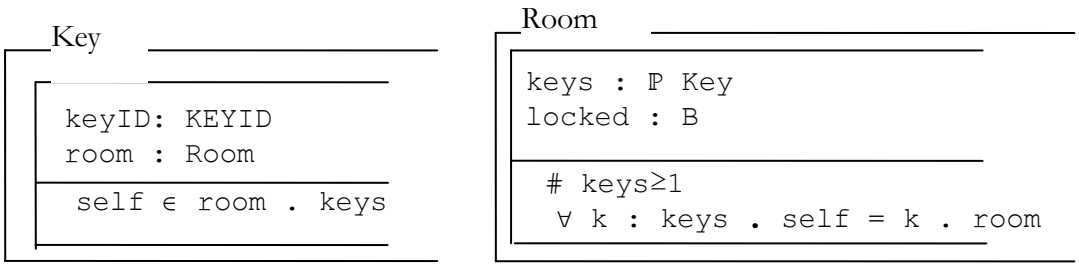


Figure 4.10 Formalizing Association: Recursive

In the third approach, a separate class schema is introduced for each association. Depending on the navigational direction and multiplicity and mandatory constraints, one or two mappings or functions, between the participating Classes is kept under its state schema with appropriate invariants. Only one instance of this Class is necessary. The traditional Z schemas, although the relationships between entities are not explicit, are usually written in this manner. It is notable that nearly all the researchers involved in formalization used this approach at least once. There are several variations of this approach (Amalio *et al.* 2003). In Duke *et al.*'s (2000) central view (they used it to cater for bi-directional navigation), separate schemas called 'Database' and 'KeySystem' are used where the first schema plays the association class role and the second one purely plays the control schema role. The third approach, as applied to the case study, is illustrated in Figure 4.11.

A common variation of the third approach is keeping two mappings to model an association (e.g. Dupuy: RoZ (2000b)) ─ see Figure 4.12a. Another variation keeps separate schemas for each mapping, for example, Araujo (1996) ─ Refer to Figure 4.12b. Although a redundant solution, in another approach, both instances and mappings are kept as attributes of both participating classes (e.g. Shroff et al. (1997)) ─ see Figure 4.12c.

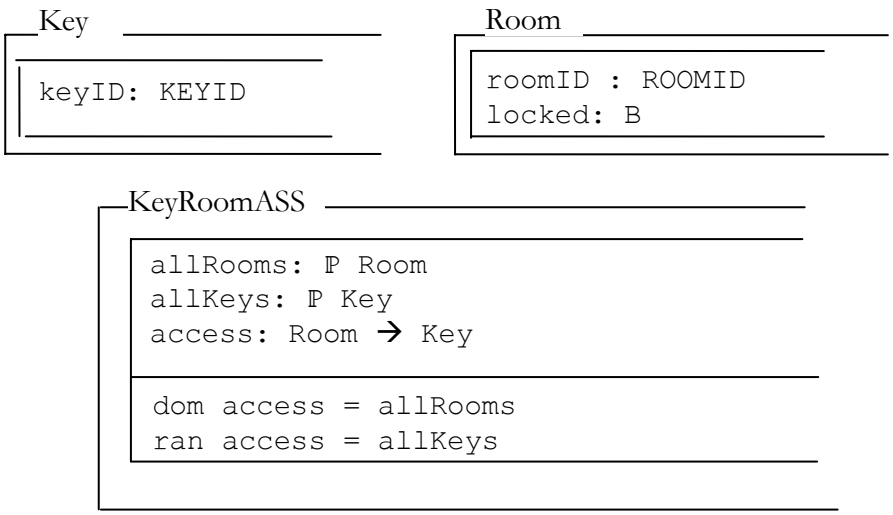


Figure 4.11 Formalizing Association: Central

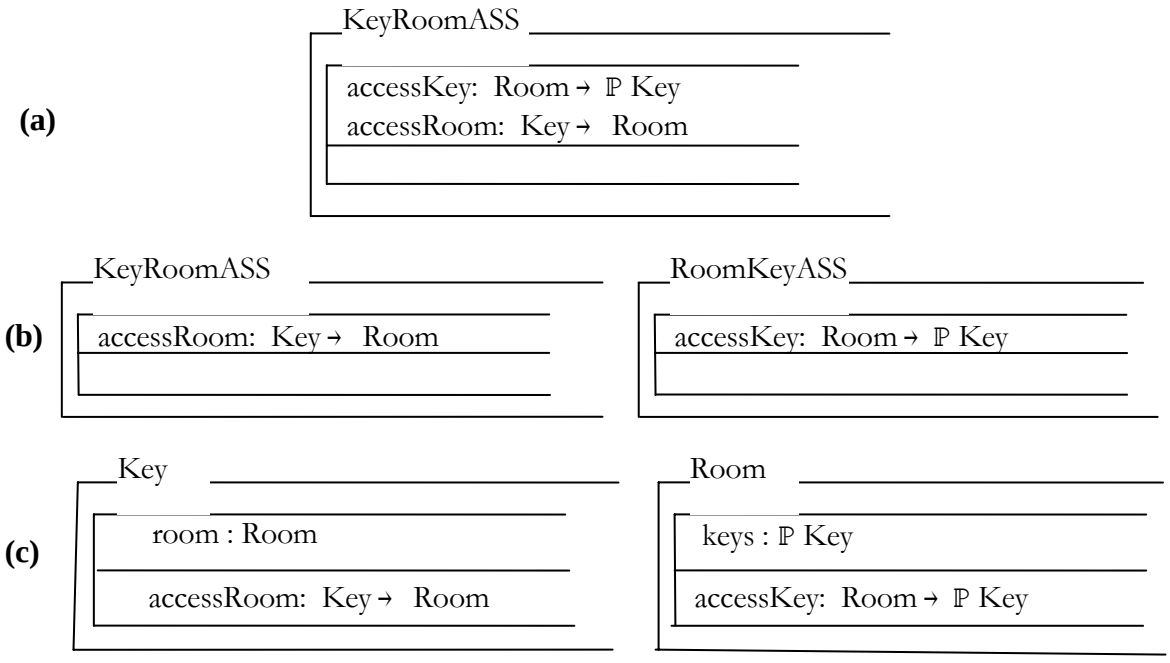


Figure 4.12 Formalizing Association: Variations

Aggregation

The navigational direction of the aggregation relationship is naturally available from whole to parts. Therefore a 'whole' object needs to know the identities of all of its 'parts'. The 'whole' object will always have an attribute that holds a set of Room objects it can open. Distinct from this, aggregation is treated in the same way as association. However, Miao *et al.* (2002), following Kim *et al.* (1999) use enumerated type variables to

differentiate three types: association, aggregation and composition. In addition to the enumeration, Dupuy *et al.* (2000b) and Kim *et al.* (*ibid.*) also use naming conventions to distinguish aggregation from simple association.

Composition

In their earlier papers, Kim *et al.* (1999) treat association as mapping, aggregation as *parts-inclusion-in-a-whole* approach and composition as aggregation with additional constraints. France *et al*. (2000) use a similar approach for composition. If the whole is removed from the system, the parts also need to be removed. Some researchers give complex proof obligation for this property (Araujo 1996). Later versions of Object-Z offer a special construct called 'containment' for this representation. Dong (1995) explains how the containment construct is useful for representing composition.

Association Class

A separate object schema is used for the association class. Similar to formalizing association, there are two main approaches for keeping the information on links. This link information may be totally discarded after including corresponding object identities as attributes, or it may be kept redundantly under the association schema (or another separate object schema may be used to keep this information). For a detailed discussion on different approaches, refer to Araujo *et al.* (1996).

Generalization

In the specification stage, inheritance is only considered as a mechanism for describing special classes from more general classes. It is an important mechanism in object orientation that provides very powerful features for the software development process, such as clarity (semantics), reusability (syntax) and flexibility (through inclusive polymorphism). If a special object behaves the same as a general object, in all circumstances, then replacing a general object by a special object will not cause any diverse effects (Goldsack *et al.* 1996). Occasionally, the behaviour of a parent may be overridden by a child. In this case, more restrictions are necessary in defining behaviour in order to allow a sub-class to be a sub-type (refer to Goldsack *et al.* (1996, Chapter 12) for more information on object types). The semantics of Object-Z and UML inheritance differ slightly. Complete overriding is not possible in Object-Z but instead, the behaviour can be enhanced in the sub classes. Since Object-Z provides inheritance as one of its

distinct features, all the above mentioned researchers conveniently used this feature to transform inheritance in UML classes to Object-Z schemas.

Polymorphism

Polymorphism literally means 'many shapes'. Booch *et al.* define it technically as *"An operation of a child that has the same signature as an operation in a parent overrides the operation of the parent: this is known as polymorphism"* (Booch *et al.* 1999, p. 64). In other words, it means that objects of different but related classes are able to serve a particular request. The relevant object and the method will be determined at run-time. Run-time issues are not pertinent in the context of specification. Polymorphism allows limited flexibility in strict typing. If a class type is associated with a variable, it may point to an Object of that Class or an Object of any other descendant Classes of that Class. Similar to inheritance, Object-Z also provides polymorphism as a distinct feature. Therefore, there is no significant difference between researchers in dealing with polymorphism.

Environment/System/Database

Traditionally, a special notation is used in the schema name to differentiate the system of objects. For the same purpose, Dupuy *et al*. (2000b) use the database schema in their RoZ project (their database schema is different because basically it is an association class). In Object-Z, it is assumed that objects are always available in the environment, and also a separate object schema is included to represent the environment. The notion of environment in Object-Z is illustrated in Figure 4.13

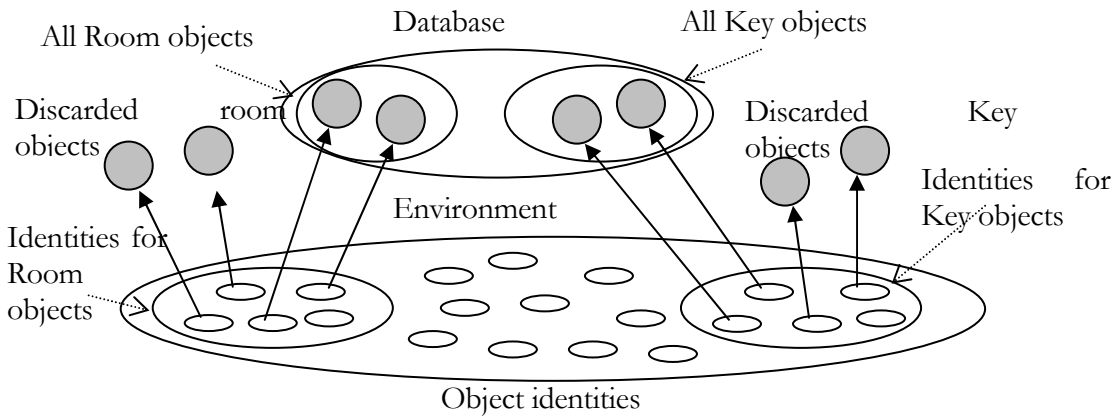


Figure 4.13 The Notion of Environment in Object-Z

All the objects will be in their initial state, if they have not undergone any transitions. A pattern like "[selection | Application]" is used in the environment schema to select an object from the environment to use it on an application (see Figure 4.8). The objects can be included in the database or removed from the database. Even after they are removed from the database, they will be available in the environment.

4.4 Summary

A constructivist (problem-transformation based) instructional strategy for CBL system is proposed in this research founded on UML to Object-Z transformation process. UML is an industry standard graphical notation for OO modelling. UML is easy to learn and use, but learning Object-Z is challenging. Object-Z is a relatively new object oriented formal notation based on the well-known Z language. Object-Z specification is a collection of Object-Z schemas. The key constituents of Object-Z schemas are state schema and operational schemas. State schemas declare the attributes, whereas the operational schemas define the operations of the class. The Visibility list specifies the accessibility options of the class features. UML specifications are used as scaffolds in the proposed strategy while teaching Object-Z.

UML is well-defined using its meta-model. Still, UML specification may be ambiguous. Formalizing UML is an important research topic in the Formal methods discipline. There have been several research initiatives reported in the literature that attempt to formalize UML with a variety of Formal methods. Primarily, the Z family of notations such as Z, Object-Z, and TCOZ are used in research towards formalizing UML models. Z is a formal notation but it is not object oriented, whereas, UML is object oriented but not formal. First of all, object oriented features need to be simulated in state oriented Z language. As a result, transforming UML to Z is very difficult. Specifically, since both notations are object oriented, transforming specifications in UML to Object-Z is considered relatively less challenging.

Translating UML models to Object-Z is useful, but still a demanding task. There are some systems designed, though not with pedagogical motivation, for transforming UML Classes to Object-Z. Surprisingly, none of the existing attempts describe a clear methodology for the transformation process. In particular, none of the approaches address

the issue of transforming UML class operations to Object-Z operation schemas. A step-by-step methodology in order to transform UML class diagrams to Object-Z schemas is proposed in this research (Chapter 9).

Chapter 5

Designing a Problem Transformation Based CBL System

5.1 Introduction

This chapter outlines the key proposals of this research for designing a constructive CBL system based on problem transformation with scaffolding. In particular, it shows that for CBL systems for complex domains active learning features could be successfully incorporated by engaging learners in suitable transformation problems. Students could be encouraged to transform some relevant known models systematically to some other complex unknown models. Scaffolding could be used to ease the complexity involved in the transformation process.

Object-Z, the exemplar domain, involves complex mathematical notations. Learning to create requirement specifications using Object-Z notation (for an undergraduate level course) involves three main stages: learning the basic syntax and semantics of Object-Z notation, learning to specify object oriented features like inheritance, polymorphism, association etc., and finally, learning to create Object-Z specifications for given case studies. The first two stages deal with primarily declarative and conceptual knowledge, whereas the final stage deals with procedural knowledge. Based on this observation, a decision is made in this study to complete the designing process in two phases. The first phase will cover the first two stages stated above, and the second phase will cover the third stage.

In this chapter (and in the next) the research relevant to the first phase is discussed. First of all, a survey was conducted to determine the effectiveness of a formal method tool as a teaching system. Section 5.2 describes the survey and discusses the result. Section 5.3 outlines the proposed instructional strategy design. This strategy, based on problem transformation with scaffolding, addresses the key challenges in learning Object-Z.

Section 5.4 describes the Four Phase Instructional model which realizes the proposed instructional strategy. The architectural design of the system "Learn Object-Z" (LOZ) which implements the proposed instructional model is illustrated in Section 5.5. The domain model is the key part of the system. Section 5.6 explains the features of the domain model in detail. Section 5.7 describes the proposed confidence based MC test strategy and the underlying marking schema. The Mentor model is described in Section 5.8. A decision table, which reflects typical tutor actions, is proposed in this section. Finally, the importance of a Learner model is established. The proposed Learner model, the linchpin of the system, is discussed in detail in Chapter 6.

5.2 Survey on a Formal Method (Z) and a Tool (Formalizer)

As mentioned earlier, there is no efficient CBL system available for learning formal methods. Although, some of the formal method tools are designed as teaching aids, to the knowledge of the author, none of them are used in real teaching. However, it is commonly accepted that teaching formal methods is difficult, without appropriate tool support (Gibson *et al.* 1998). In fact, tools like Z/EVES (Saaltink 1997) and Formalizer (Flynn *et al.* 1989) are freely available and widely used in Universities to support formal method courses (especially for Z). A study was conducted initially to investigate whether any of these tools could enhance (or hinder) the learning of formal methods. Formalizer was selected for examination as it was used at that time (2004) in Massey University with the formal method course[4]. Formalizer is mainly useful for preparing Z specifications. 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 mainly to test the effectiveness of Formalizer in learning the Z specification (see Appendix A). 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.

The following three research hypotheses were considered critical to the investigation:

- Z is difficult to learn

[4] Currently, there are no full-fledged tools available for Object-Z notation. A type checker Wizard, and a graphical editor Moby/OZ are mentioned on the Object-Z web site (Graeme 2007)

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

Subjects of this survey 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 students returned the completed questionnaire. Except for one person, all the students answered all the questions. The raw data and relevant calculations are given in Appendix A. In general, the evaluation revealed 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 five point scale as relevant.

The main observations are:

- More than 40% report that Z is difficult to learn
- Only 3% report Formalizer motivated them to learn Z
- More than 60% report that Formalizer is difficult to learn.
- Nearly 70% thought that the help facilities of Formalizer were useless.
- More than 90% of the subjects reported that Formalizer is annoying to use.

Novice users were found to be more exasperated by the interface of Formalizer. The features such as type-checking that experienced users consider very useful were considered difficult by novices. Only about 25% of the subjects considered that some special features in Formalizer were really useful. As expected, 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 stated that an on-line tutorial with proper feedback would help them to learn Z notation while using Formalizer.

The results suggest that a considerable proportion of students found that learning formal methods is challenging in the current environment. The tool support was not sufficient; moreover, it had negative effects on the learning process. The situation could be improved by using a software application which provides an effective learning environment with

suitable artefacts such as context-sensitive feedback, and visual aids to help comprehend the complexity of the structure. The system should also be flexible enough for a trial and error learning process. The following sections will progressively describe the research related to designing an easy-to-use and enjoyable CBL system that alleviates the difficulties associated with learning the Object-Z notation.

5.3 Designing an Instruction Strategy

As a first step towards designing a CBL system for a complex domain (e.g. Object-Z), a suitable instructional strategy needs to be formulated. In order to devise an instructional strategy, the defining features of the domain need to be analysed. Chapters 3 and 4 discuss important characteristics of formal methods in general and Object-Z in particular. The next section points out some specific characteristics of the problem domain that leads to the instructional strategy proposed in this research.

5.3.1 Problem Domain

First of all, Object-Z is an object oriented formal language. Anticipated users of the proposed CBL system will be university undergraduates and software practitioners. Academic knowledge has second-order characteristics, and depends heavily on symbolic manipulation (e.g. UML diagrams, mathematical symbols), whereas everyday knowledge is of first-order and does not require any interpretation. Laurillard make a case for formalisation in academic learning: "*The argument against the primacy of formal knowledge, advanced by Gilbons et al, and by Brown and Duguid, thus comes full circle to an acknowledgement that without the process of decontextualisation, and formalisation, knowledge remains situated and uncommunicable"* (Laurillard 2002, p. 23). In line with her observation, it may be noted that Object-Z, being a formal notation, demands a higher level of formalization even in the early stages of the learning process. Therefore, a case-study-based instructional approach (mingled with formalized content) will be used in the proposed system in order to help learners abstract the relevant concepts from hands-on experience. The case study approach will be discussed further in Section 5.2.3.

Research identified four key reasons for the difficulties associated with learning formal methods: insufficient mathematical ability, complex notation and structure, lack of motivation, and finally inability to abstract details (while learning mathematical notations simultaneously). The second and third issues have been addressed by systems like Zbrowser (Mikusiak *et al.* 1995) and (ZAL/ZED (Morrey *et al.* 1993). The third issue, lack of motivation, could be alleviated by incorporating a Refinement unit that boosts extrinsic motivation and provides an active learning environment. The next section will describe the proposed Refinement unit.

The final problem in the list is mainly addressed in this research. Learning Object-Z is significantly more challenging as it demands, in particular, two higher levels of abstraction processes concurrently: Object Oriented Abstraction and Mathematical Abstraction. The first process involves transforming real world facts and processes into a collection of models so that the models can be easily converted eventually into computer programs. The second process involves representing the key requirements and constraints on those models using well-defined mathematical symbols. Section 5.3.2 will describe the proposed solution for the abstraction problem.

5.3.1 Refinement Unit

Students often lack motivation to learn formal methods in the early stages as they just perceive it as a notation with complex structures and mathematical formulas, and fail to recognize its advantages. The impact of motivation in learning has been well analysed in educational psychology and later in cognitive science disciplines. In the context of learning formal notations, Crocker puts it thus: *"Students and software developers need to be provided with positive feedback at regular intervals to motivate them to continue working hard"* (Crocker 2003, p.92). Tertiary students are usually extrinsically motivated. The fourth form of extrinsic motivation in self-determination theory, is rather higher, and may be achieved by the learners whose goals are fully identical with how they perceive the system (Ryan *et al.* 2000). Finney states: *"Often* [students] *do not immediately see the relevance of* [formal method courses] *to what they are learning in the rest of their program or their intended profession"* (Finney 2003, p. 97). To maintain at least the third form of extrinsic motivation, the system should convince learners that their effort in

learning such a complex mathematical notation is useful and matches their long term goals.

In order to make the students feel that learning Object-Z is useful, a software animation tool (Refinement unit), which could demonstrate the tangible benefits of the formalization process, is incorporated in the CBL system. This Refinement unit translates Object-Z specifications to Java programs, automatically or semi-automatically with necessary interaction with the learner (where the system may provide scaffolding). These Java programs may be executed with selected test cases and the output may be analysed. The Object-Z specifications may be altered to see its impact on coding. The learners will then realize that the Object-Z notation is really useful for creating a precise specification, and therefore does have a significant impact on the final product.

The Refinement unit plays two roles in this system. Particularly, in the early stages of the learning process it serves as a tool that extrinsically motivates learners; and secondly, it provides a constructive learning environment based on problem transformation with scaffolding. The next section will describe the proposed strategy to address the 'abstraction' problem in learning Object-Z notation.

5.3.2 Scaffolding Abstraction

Abstraction is an important intellectual tool composed of two processes: elaborating essential detail (depending on the context) by transforming it into suitable models, and generalizing the unnecessary information into its minimal supportive form. The process of creating formal specifications from problem descriptions, particularly for Object-Z specifications, demands a very high-level abstraction ability (Duke *et al.* 2000).

Scaffolding

Being a higher level intellectual ability, abstracting a real problem requires significant ability to group material into familiar chunks (to form relevant schemas) effectively. Therefore, allocating resources in the working memory for learning a complex mathematical language in order to represent the key characteristics of abstraction (while engaged in the abstraction process itself) is a challenging task. To facilitate relevant schema acquisition, the instructional technique should allow the user to learn the syntax

and semantics of Object-Z notation independent from the abstraction process. Students may initially learn how to apply the mathematical notations to represent the abstracted constraints in the application domain before learning the underlying abstraction process. Ultimately, however, the learner should be able to model the structure of the system and specify the data and functional requirements straight-away in mathematical notations. To aid this learning process, scaffolding is used in the proposed CBL system.

Traditionally, the scaffolding technique (see Section 2.6.1) is used for training in complex physical skills. Collins *et al* (1990) proposed a model that uses scaffolding for learning complex cognitive tasks. In this model, to train a novice in a complex cognitive task, coaches initially complete the difficult processes and allow the trainee to concentrate on a portion of the task that can be comprehended within their working memory limits.

Role of UML in Scaffolding

In the proposed system, learners are initially supported by the UML models in the abstraction process, and are freed to concentrate on learning mathematical notation to represent those abstractions. UML models are semi-formal models that deal with a considerable level of data and functional abstraction. In this way, the working memory of the learner may be used to its full capacity to integrate the mathematical notations with the available key abstractions. A learner may create Object-Z specifications by transforming the relevant UML model. This task is relatively easy compared to creating Object-Z specifications from textual descriptions alone. Gradually, the supportive UML model will include fewer and fewer abstractions, and the learner will be encouraged to perform more and more data and functional abstractions expressing them in mathematical notations. Eventually, the UML support will be totally withdrawn and the learners should be able to perform abstractions themselves and represent the key characteristics in mathematical notations. In other words, ultimately the learner will be able to produce Object-Z specifications from a given textual description.

Problem Transformation

The learner will be encouraged to transform the UML models to Object-Z models in steps. The system will provide necessary and sufficient support for using mathematical notation and Object-Z structure during transformation. This approach provides a

constructive learning environment based on problem transformation. The system may provide an appropriate online environment with scaffolding for a step-by-step transformation process, or the learner may be encouraged to do the transformation off-line.

Using a semi-formal model for learning formal notation is not a totally new idea. Barden *et al* (1994) uses E-R diagrams to teach Z notation. Sun *et al* (2001) also claim that their system may be used for learning Object-Z or UML. Naturally, the students will tend to use some semi-formal model (with which they are comfortable) in the intermediate steps to create a relevant formal model. The E-R diagram only models data abstractions, whereas the UML models can accommodate data and functional abstractions in encapsulated manner.

It is expected that the potential users of the proposed system will have sufficient knowledge and experience in using the UML models for requirement specifications. The UML notation is selected in this research as it is the de facto standard for Object Oriented semi-formal modelling, and is also recommended in the ACM Software Engineering curriculum for basic modelling and design (ACM-SE 2004, p. 59). Many universities now teach UML in the first or second year. The formal methods, if included, are usually taught after teaching UML models.

5.3.3 Case Study Approach

Situated cognition indicates that learning will be successful if it is carried out in a related context. Virtual reality systems are one of the most extreme CBL systems that could facilitate learning in such a manner. At a moderate level, the case-study based instructional approaches, which are a form of problem based learning, are useful in creating the required context in many situations (Shute *et al.* 1995). The case-study approach is used in this research in both phases. The nature of the subject domain, formal methods, naturally demands case-studies for its explanation. Almost all the books on formal methods extensively use case studies for modelling complex situations. Dean *et al* advise that the case studies should be selected carefully, *"relatively simple case studies* [that fail to highlight the need to use formal methods at early stages in the development]

tend to be dull and uninteresting" (Dean *et al.* 1996, p. 101). They further add *"Case studies are needed which explore the possibilities more fully" (ibid.)*.

The proposed system uses worked examples based on appropriate case studies to illustrate the underlying concepts, notations and techniques. The system helps learners to concentrate on the selected concepts by hiding irrelevant complex processes in the case studies. The formative assessments are carefully designed in order to identify potential misconceptions related to the case studies. Feedback may also include portions of relevant case studies. The Refinement unit provides relevant Java code and test schemas for the selected case studies.

In summary, before devising the architecture of a CBL system for a complex discipline, the domain needs to be carefully analysed in order to design the fundamental strategy for instruction. For example, Object-Z is a formal notation. Inability to abstract detail is found to be a major drawback for the students in learning this discipline. Scaffolding abstraction using UML models could resolve this issue. In the meantime, encouraging learners to construct Object-Z models from UML for carefully organized case-studies provides active learning opportunities. Integrating an efficient Refinement unit in the system could help to keep the learner's extrinsic motivation high. The next section describes the proposed instructional model that facilitates the instructional strategy.

5.4 A Four Phase Instructional Model

In general, the instructional strategy proposed in the previous section is based on constructivism. Constructivism is not merely a learning theory, but a whole philosophy. Unfortunately, practical constructivist instructional systems are, as Dick (1991) points out, costly to develop and require technology to implement. To cope with the technology and educational environments, the proposed instructional model needs to include techniques based on cognitive and behaviourist learning theories.

In line with the cognitive apprenticeship instructional paradigm devised by Collins *et al* (1990), an instructional framework called Four-Phase Scaffolding Instructional Model (FoPSI) is proposed in this research. Their original model is highly suitable for learning complex procedural tasks, and contains six aspects: modelling, scaffolding and fading,

coaching, reflecting, articulating and exploring. The FoPSI model consists of four phases: Pre-conditioning, Modelling, Building, and Exploration.

5.4.1 The Four-Phases of FoPSI model

The learning material in the proposed system (for Phase I) is organized as a collection of concepts in a hierarchical order (see Section 5.5). For each concept, the FoPSI model encourages the learner to go through four phases: Pre-Conditioning, Modelling, Building, and Exploring. Learners are given facilities in the last two phases to reflect on their learning process and also to articulate their learning progress, possibly with peers and mentors.

Pre-Conditioning:

The pre-conditioning phase is not included in the original (Collin *et al*'s) model. Constructivist learning theory suggests that learning will be effective if the learner is in a state of cognitive readiness, that is, their past relevant knowledge is activated and their cognition is conditioned so that they will be able to build new knowledge based on their past knowledge. In particular, this phase is included explicitly in the FoPSI model as the knowledge in pre-requisite disciplines is essential for the instructional techniques discussed earlier. For example, before presenting learning materials relevant to an Object-Z concept, the learning materials related to the corresponding object oriented concepts and UML notations will be introduced to the learner.

Modelling

The Modelling phase in the proposed approach is equivalent to the original phase 'modelling' in the original work. The learning materials relevant to the current domain (e.g. Object-Z) are presented. The learner may examine some worked examples. Suitable illustrations will be provided with examples to enable the learner to abstract the relevant knowledge from the given context. In this stage, the learner develops the general idea about the learning domain based on pre-requisite disciplines. For example, at the end of this phase, if suitable support is given in UML form, the learner will be able to create relevant Object-Z specifications.

Building

The Building phase is the most important part of FoPSI. The related aspects in the original model are scaffolding and fading. Scaffolding is normally used in CBL systems for learning complex procedural tasks; here it is used for learning conceptual knowledge. As explained before, initially, learners are given help with the abstraction process. For example, UML diagrams will be provided to supplement textual descriptions. Gradually, the UML diagrams are withdrawn and learners are encouraged to perform abstractions on their own and represent those using mathematical notations. Eventually, the learner will be able to create Object-Z specifications from textual descriptions alone.

For each concept, a series of scaffolding levels are designed with decreasing levels of UML support. For each scaffolding level, a set of problems are included for assessment. The system will provide a suitable environment for active learning by engaging the learners in constructing Object-Z models from given UML constructs. Feedback will address potential misconceptions and is carefully designed to support the scaffolding process further. The learners may compare an Object-Z specification to related Java code. They may test it and feel the impact of their specification on the final program. Learners build their own understanding of the topic by solving increasingly challenging problems.

Exploration:

The Exploration phase in FoPSI is more intensive than that of the Collin's original model. As Collins *et al* put it, *"teachers need to encourage students to explore questions teachers cannot answer, to challenge solutions the "experts" have found- in short to allow the role of expert and student to be transformed"* (Collins *et al.* 1990, p. 490). FoPSI encourages learners to go beyond that. Learners are encouraged and empowered to critically analyse and challenge what they have learned so far and to learn more through other means (e.g. web, email etc) This phase will be mainly carried out through independent exploration and discussions (with peers, mentor and other relevant sources).

Research in cognitive science has proposed several learning theories that stress exploration, which basically refers to the processes of finding multiple positive evidence for some recently learned items. By this process, the learner's cognitive search paths will

become strong. Unfortunately, the paths may get so strong that any alternatives could be rejected instantly without being analysed. What is proposed for exploration in this research is considerably different. In a cognitive learning view, instead of cementing a belief in one's cognition, it is proposed that the learner shall always analyse and challenge the current beliefs critically, and keep their cognitive search paths both active and accommodative. In other words, beyond the situated learning view, the exploration phase encourages the learner, if necessary in the light of different contexts, to question the validity of eventual abstraction. The next section will describe the proposed architecture of a CBL system that incorporates this instructional model.

5.5 Overview of LOZ

A CBL system called LOZ is proposed for learning Object-Z notation (for phase I), which facilitates the instructional strategy discussed before. Figure 5.1 gives a high level view of the overall architecture of the system. Except for the Refinement unit, other components can be found in any typical ITS (Hartley *et al.* 1973). The Refinement unit helps to preserve student motivation. It may automatically generate programming code or it may provide an interactive environment where the learner may construct the code step-by-step with the system's assistance.

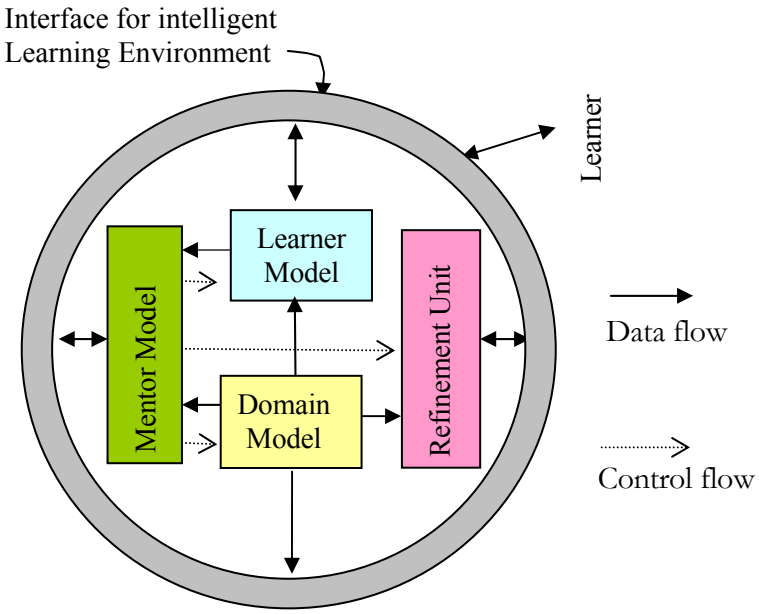


Figure 5.1 Architecture of LOZ

In principle, the pedagogical approach of this system is constructivist. Basically, it provides a guided environment where learners build their own knowledge while interacting with the system. Though the system can provide various levels of assistance tailored to individual learners' requirements, they may simply ignore it. Users may, if they wish, take full control of their own learning process and let the system provide useful guidance only when required.

In order to give effective guidance tailored to an individual learner, the system needs to have some knowledge about the learner. The Learner model tries to measure the learner's competence level in understanding current concepts, general learning ability, traits etc. The learner is empowered to view and alter their model. In many cases, learners' responses are analysed in a Mentor model in order to provide appropriate feedback and curriculum sequencing. For this purpose, the Learner model provides the required information about the learner and the domain model provides subject specific knowledge.

A potential user of LOZ is expected to be a software engineering student or a software practitioner. The user is expected to have adequate knowledge in Object Oriented Concepts, UML specification, and Discrete Mathematics. In particular, the UML knowledge is essential as it will be used in scaffolding. Knowledge of Z notation, though useful, is not necessary. For the same syntax, the semantics of Z and Object-Z may significantly differ in some cases (e.g. Delta operator in operation schemas (Duke *et al.* 2000)).

Each component of LOZ, except the Learner Model, will be discussed in the following sections. The Learner Model is discussed in Chapter 6. Incidentally, the division between the components and processes are not strict. For example, feedback and assessments are discussed under Domain model; however, the selection process is the task of a Mentor model using the information provided by a Learner model.

5.6 Domain Model

The first step in designing instruction based on scaffolding is identifying the appropriate scaffolding steps and designing suitable lesson material that keeps seamless continuity between those steps (Turnbull *et al.* 1999). The learning material associated with the

domain knowledge in LOZ is structured in several lessons. Each lesson has several topics and each topic has several main-concepts. A main-concept may have several sub-concepts. A pre-condition concept is associated with each main-concept. The pre-condition concepts are UML and/or Object Oriented concepts related to the corresponding main-concept in Object-Z. Several exploratory concepts are also assigned to each main-concept. The exploratory concepts help the learner to critically analyse the main-concept in alternative views. In the structural view of the domain, a preferred order in which these topics could be learned is given in a hierarchical network (Figure 5.2). A leaf-concept usually corresponds to various supporting atoms. An atom may be, for example, a video-clip, an audio-clip, textual description or an URL, etc. This type of hierarchical curriculum organization is common in educational environments.

This curriculum organization in Figure 5.2 gives the part-of (aggregation) relationship – the branches are part-of the root elements. In particular, each sub-concept is associated with a series of mental states (the notion of mental state with respect to this research will be discussed in the next section). In other words, the domain knowledge in the system is organized in a hierarchy of series of mental states. For each sub-concept, there will be a basic, final and some intermediate mental states. The learning outcome of a sub-concept is attaining considerable strength in the associated final mental state.

Figure 5.3 illustrates an example for a sub-concept and the series of the related mental states. After learning the material relevant to a sub-concept (at the end of the modelling phase in FoPSI), the system assumes that the learner has considerable strength in the basic mental state. Thereafter, the system will provide a suitable scaffolding environment for the learner to gradually proceed to gain sufficient strength in the final mental state.

The intermediate mental states are not arbitrary. They are carefully selected so that the related learning material should be sufficient to challenge the student while providing necessary support.

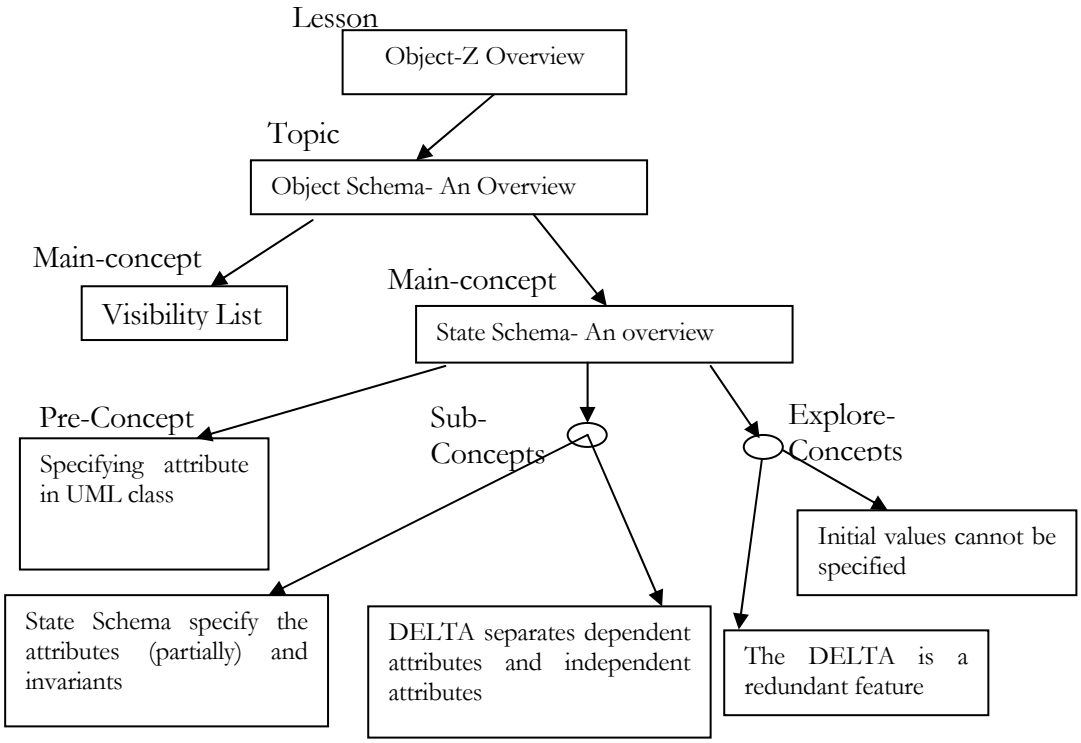


Figure 5.2 Lesson Organization– An Example

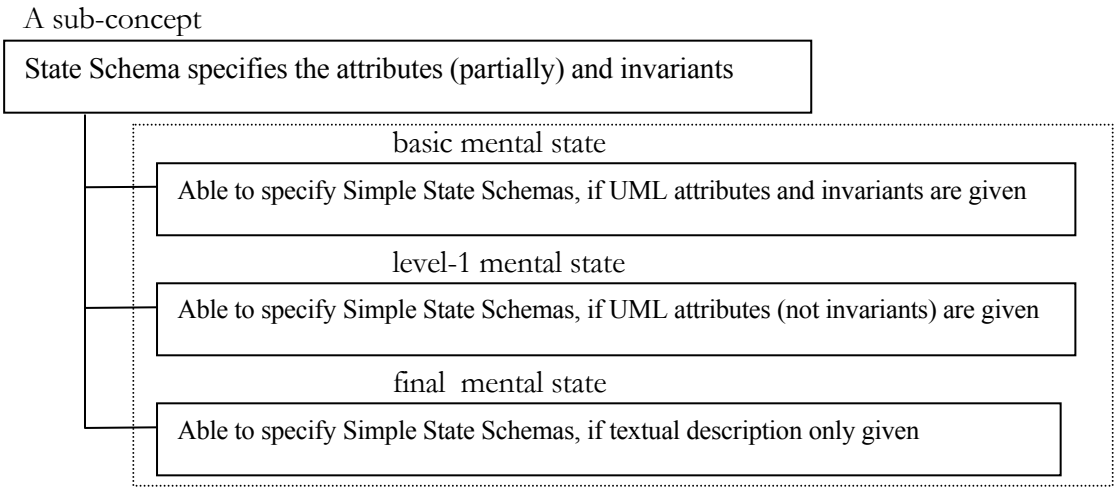


Figure 5.3 A Sub-concept and related Mental States

5.6.1 Mental States

The term 'mental state' is used widely in psychology and is usually associated with the state of mind of a human being. It is defined as *"a mental condition in which the qualities of a state are relatively constant even though the state itself may be dynamic"* (WordNet 2005). And another related term 'mental object' (or 'cognitive content') is defined as "*the*

sum or range of what has been perceived, discovered, or learned" (WorldWeb 2005). In this research, the term 'mental state' is interpreted differently. A person will be in a certain 'mental state' (or 'cognitive state') if they have already constructed sufficient knowledge (and/or skills) which enable them to perform a certain mental or physical task. Though not defined explicitly, Ragnemalm (1995) uses the term 'mental state' in a similar context. Note that the notion of mental states is not rigidly defined in this study. The qualifier 'simple' in the mental state "Able to specify simple State Schemas, if UML attributes and invariants are given" (Figure 5.3) is vague. This is not unusual for human beings. Our every-day language consists of many such vague qualifiers. In fact, learners and mentors are familiar with these types of interpretation.

As stated before, a main-concept consists of a series of sub-concepts, and each sub-concept is associated with a series of mental states. Mental states are to be carefully selected as each of them in the series is associated with a scaffolding stage. For a particular sub-concept, after finishing the modelling phase (at the beginning of the building phase), the learner is expected to be in the lowest mental state in the series. Gradually, they will move to the highest mental state at the end of the building phase. The number of intermediate mental states and their types will determine the intensity of the scaffolding process. Too much or too little scaffolding may undermine the learning process: too much support may cause boredom due to lack of challenge and too little assistance may cause frustration. Nevertheless, determining the necessary and sufficient level of scaffolding is challenging as it varies between learners. A lower level learner may require fine-grained levels of mental states. Even for a single learner it may vary between different situations, moods etc. A strategy may be designed to predict the appropriate scaffolding level automatically (number of steps to be jumped) based on the past performance of the learner. Such a strategy will be illustrated later in Chapter 6 under Learner modelling.

Obviously, the mental states in the series related to each sub-concept are heavily dependent on each other. Figure 5.4 shows (within the dotted square) the causal relationships between the mental states related to the same main-concept. Particularly, the strength of a mental state in a series will definitely have an impact on its preceding mental states. For example, if learners can specify state schema from a textual description, they will find it very easy to do so with the aid of UML. Conversely, the strength in earlier

mental states in a series will have some impact on the subsequent mental states in a series (this is shown as arrows in Figure 5.4). Between any two series of the same main-concept only the basic mental states are considered dependent. In another view, simple arrows show pre-requisites and dotted arrows (outside the square) show composition relations from branches to nodes. These simple causal relationships will be used in Learner modelling (Chapter 6).

Incidentally, the mental states associated with different series of different main-concepts may also depend on one another. However, to avoid a complicated causal network, only limited causal relationships between mental states are considered in this study. This assumption is weak, but agrees with the notion of locally intelligent models (McArther *et al*, 1999), based on which the Learner model is built in Chapter 6.

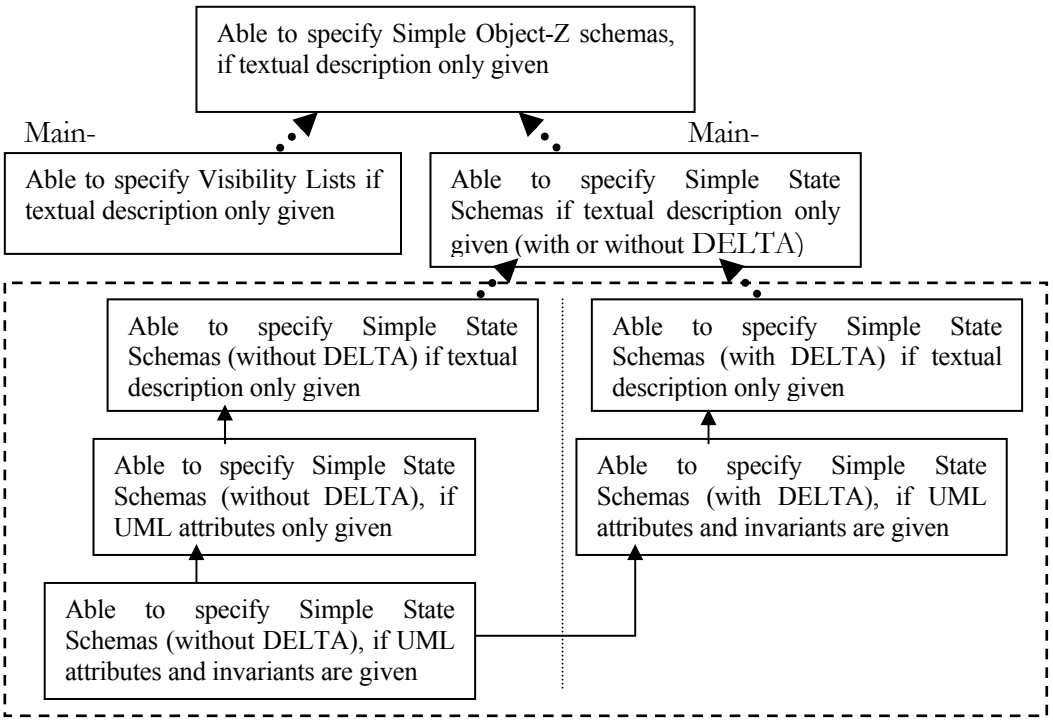


Figure 5.4 Mental States- Causal Relations

The strength of the causal relationships in Figure 5.4 (or prior probabilities, if a Bayesian Network is considered) may be suggested initially by domain experts (or empirical studies). These values may be tuned later by machine learning methods.

As Murray *et al* put it,

> *"However, an encouraging result from prior research is that Bayesian systems are often surprisingly insensitive to imprecision in specification of numerical probabilities (Henrion et al. 1996) and may be accurate enough to infer the correct decision even if some of their assumptions are violated (Domingos et al. 1997), so that the precise numbers may not always be necessary"* (Murray *et al.* 2000, p. 161).

For example, it may be assumed that if a learner is able to specify a simple state schema without DELTA, they will also, after learning the relevant material, be able to specify a simple state schema with DELTA with 0.8 probability (if textual problem descriptions are only given in both cases). The symbol DELTA in a state schema is used to separate the dependent and independent attributes. Weights may be assigned for composite relationships also. For example, the fact that a learner is able to specify a simple state schema without DELTA contributes to the fact (with certain weightings) that they are also able to specify a simple state schema (with or without DELTA - both from a textual description: see Figure 5.4).

During the second phase of the FoPSI (modelling), learning materials related to a sub-concept will be presented. After a learner has assimilated the concept, the system assumes that they have attained a considerable strength in the corresponding basic mental state (Level-1 scaffolding). At the beginning of the third phase (building), the system performs formative assessments to make sure that the learner can perform certain tasks related to the basic mental state and can move to the next scaffolding level. Multiple Choice tests are chosen for formative assessment in this research. The rationale behind this choice will be discussed in the next section.

5.6.2 Multiple Choice tests for Formative Assessment

Multiple choice (MC) tests are used in the first phase for assessing the mental states of the learner in a systematic way, and they provide pertinent information to the system so that the learners can be guided in a step-by-step fashion. The essay-type questions, though better for testing higher level cognitive states in Bloom's taxonomy (1956), are not feasible for formative assessment in this research. First of all, being a graphical notation,

creating and editing the Object-Z specification demands a sophisticated graphical editor. Moreover, being a formal notation, interpreting Object-Z specification requires complex processes such as parsing, syntax checking and type checking. Rather than being distracted by these complex and time consuming activities, focusing on the pedagogical aspects related to the semantics of Object-Z notation is considered essential at this stage.

The MC test is considered unsuitable to assess higher level cognitive abilities such as synthesizing or evaluating in Bloom's taxonomy. In particular, for programming language instruction and similar procedural domains, some research shows (Simkin et al. 2005), essay-type tests are better than the MC tests in measuring the student's ability to solve real world problems. Nevertheless, the focus in the first phase is mainly on learning declarative and conceptual knowledge, and therefore, the usage of MC tests for formative assessment in the scaffolding process is justifiable. However, the traditional MC tests are very limited and need to be tuned to suit the requirements of this research. Section 5.7 discusses the proposed confidence-based MC test scheme designed in this research for this purpose.

The interaction bandwidth in traditional MC test is very limited (usually a single mouse-click). Students need to select only one of the given options as the correct answer. On the other hand, for critical response assessments students may give any input (limited by only the interface). If the system includes rich natural language processing systems (and other notation parsers if necessary), a fine-grained domain model, and a knowledge tracing learner model, the system could minutely assess the knowledge levels of students, and would provide appropriate feedback tailored to individual learners. The benefits of using critical response type assessments for learning procedural type knowledge are illustrated in Chapter 9.

5.6.3 Misconceptions and Feedback

The learner may need to go through several tests and relevant feedback before acquiring the required strength in a mental state. The distracters in the tests are associated with potential misconceptions related to the current sub-concept. Significant empirical studies are necessary to identify potential misconceptions related to each sub-concept. Different levels of feedback materials are designed for each misconception (associated with distracters). Suitable worked examples and discussions are also associated with each

misconception and question. Java code (and test schemas), if applicable, are also available for relevant questions, examples, and distracters (see Section 5.8). Figure 5.5 gives an overview of the domain model in LOZ. When the learner is ready for the next scaffolding level, the system selects the next highest mental state in the series that falls within their zone of proximal development. Progressively, in this way, the learners may be guided from the basic mental state to the highest mental state, using formative assessments and relevant feedback.

As mentioned earlier, in this research, the UML models are used as scaffolds. Initially, in the MC tests, learners are given UML specifications and required to select corresponding Object-Z specifications (learners have to work offline to construct relevant models). In other words, they are initially freed from performing abstraction and encouraged to concentrate on applying mathematical notations to represent the key abstractions provided in the UML specifications. Gradually, the UML help in the tests will be withdrawn, and therefore, eventually, the learner will be able to perform abstractions as well as be able to apply mathematical notations to represent those abstractions.

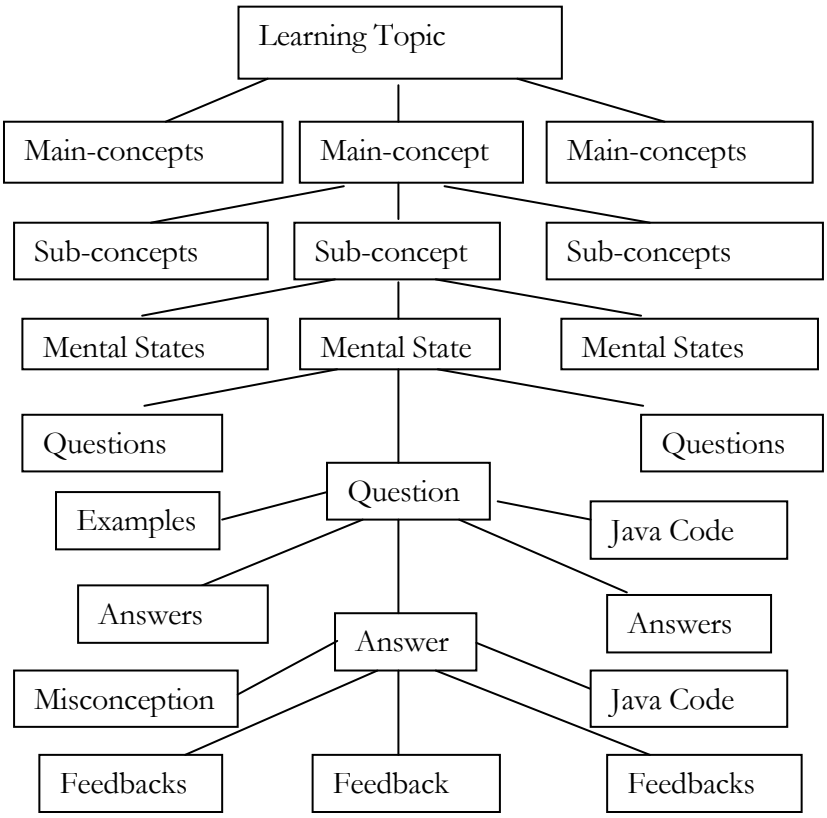


Figure 5.5 Overview of the Domain Model in LOZ

In summary, the Domain model plays an important role in our design. Concepts to be learned are organized in a hierarchical format. A main-concept has a series of sub-concepts. A series of mental states are associated with each sub-concept. Each sub-concept is associated with some misconceptions. Different levels of Feedback are designed to address each misconception. Each mental state is related to a scaffolding stage. The expected outcome of learning each concept is attaining the related final mental state. MC tests are used to assess a learner's strength in the current topic. Distracters are designed to identify potential misconceptions.

5.7 MC Tests as Scaffolding Blocks

The MC tests play a key role in the scaffolding process: they not only assess the learner in each scaffolding stage, but also identify their potential misconceptions. Different levels of feedback are designed to address each misconception for different levels of learners.

5.7.1 Drawback of Traditional MC test

Although the traditional MC test is a simple yet efficient formative assessment tool, this version is not flexible enough for a learner to express their state of knowledge. The interface of the traditional version is severely limited; it only allows the learner to mark one tick (to select a correct answer). If a learner answers a test item correctly, it cannot be concluded that they are strong in the corresponding mental state. A correct answer may be just a lucky guess, an informed guess, interacting misconceptions (for example, a mixture of two misconceptions may result in a correct answer), or mastery. Similarly, if a learner selects a wrong answer, in most cases, the system assumes that they have a certain misconception and try to treat it. This is also a weak assumption. The learner may select an answer by just a blind guess, as a result of an informed guess (eliminating wrong answers, grouping most suitable answers or by both), just a mistake, or due to the associated or some other misconception. A weak but lucky person should not be treated as an expert; it is not good for their learning process. In the meantime, those who have partial knowledge should be identified and treated properly.

There are formal techniques available for making decisions under such uncertain situations (see Chapter 6). However, in traditional MC tests, these techniques are severely

limited due to insufficient evidence (because only one answer can be selected). The testing system should allow the learner to express their level of understanding about the subject matter (relevant to the test item and the distracters) more precisely. This drawback of the traditional version hinders the whole scaffolding process as the state of the learner cannot be identified properly; and therefore, appropriate feedback materials cannot be presented to address the need of the learner. As a remedy, a confidence-based MC test schema is designed in this research so that the learner can express their knowledge on the concept under consideration more precisely.

5.7.2 Designing a Confidence-Based MC Test Schema

Confidence Based Marking (CBM) method (discussed in Chapter 3), to a certain degree, allows the learner to express their knowledge level in the test topic (Gardner-Medwin 1995). However, his method allows the learner to select only one answer that they think correct (though with various degrees of confidence). Since the learner cannot express anything on other answers the rate of potential misconceptions related to the unselected answers cannot be totally identified. Therefore, this method is not very suitable for this research. Building knowledge in the scaffolding process is highly dependent on rich and relevant feedback.

Although an imprecise classification, the Coombs' method (discussed in Chapter 3), which requires the students to mark all the wrong options, is claimed to be able to identify five levels of knowledge on any MC test item: full knowledge (3- all and only the incorrect options marked), partial knowledge (1 or 2- some incorrect options marked), partial misinformation (-1 or –2 - some incorrect options marked, but correct option is also marked), full misinformation (-3 – only the correct option is marked), and absence of knowledge (0 – not answered at all or selected all options) (Bradbard *et al.* 2004). However, Coombs' method does not have any feature to specify the degree of belief in individual options; and therefore, it cannot explicitly identify the degree of misconceptions associated with different distracters (Coombs *et al.* 1956). This requires a more precise and explicit mechanism for learners to express their belief against each answer option of a test item.

Based on both methods, the CBM method by Gardner-Medwin and the Elimination method by Coombs, an answering interface (Figure 5.6) has been designed for this research in order to enable the learners to indicate their degree of belief precisely. Learners may express their belief not only on their best answer but also, if they wish, on each other answer option. Learners may just select 'not sure' or express their belief on either band: correctness or incorrectness (obviously not on both). Initially, all the slide bars on 'incorrect' side and 'correct' side will point to 0%, and all the 'not sure' radio buttons will be selected. If a learner is 100% sure that a particular answer is correct or wrong, they just need to click the related radio-button. In both cases, the related slider will move to point 100%. Otherwise, learners may specify their level of belief using the relevant slide bars for the related answers. In this case, the MC test may be considered as a group of true/false category tests (where the 'false' option usually has higher chance in a group). Based on the degree of misconceptions, the system is now able to provide highly relevant feedback

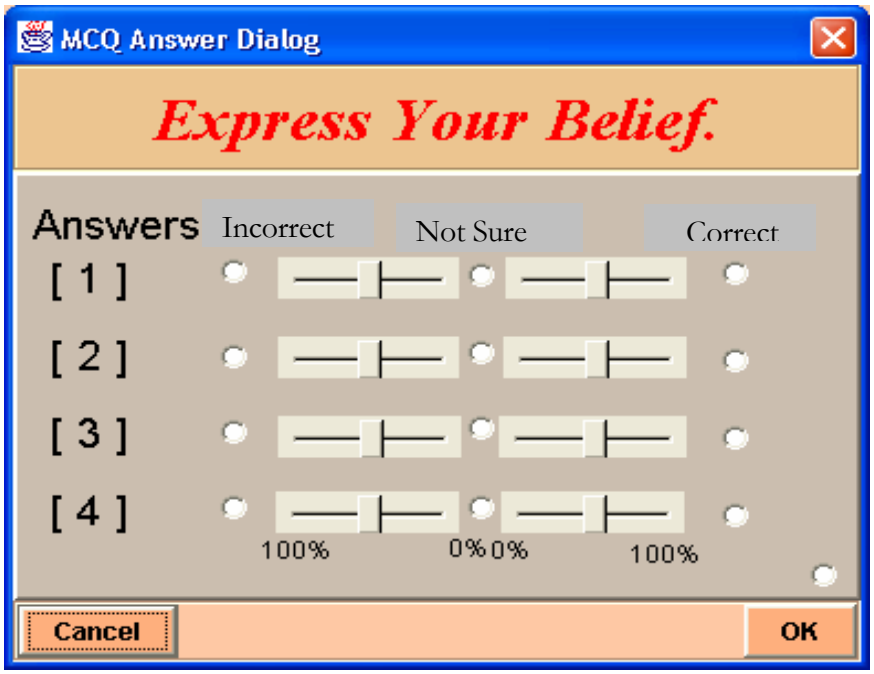


Figure 5.6 Interface for Marking Confidence

Marking Schema

In the traditional version, performance in a test item may be considered as a binary variable: right or wrong. But in a CBM method, it may not be either right or wrong. It

may be both right and wrong in different degrees. It may be necessary to measure the knowledge level on a test item (or degree of strength in certain mental state) as a single numeric value in order to select certain pedagogical actions (this issue will be discussed in detail in the next section). For example, this measure may take any value between 100 (for exactly correct) and 0 (totally wrong). CBM methods can explicitly measure different levels of misconceptions related to each answer options, but cannot explicitly provide a single numerical value for the performance on a test item.

A marking scheme that gives a single numerical value as performance measure is proposed in this research. The discussion here is restricted to the MC tests with four options and one correct answer. Gardner-Medwin employed high negative marks for strong misconceptions (for level 3 confidence, -6 marks if the answer is wrong, but only 3 if it is right), which may work well in the medical profession as improper diagnosis is critical ('not sure' option may be better than wrong diagnosis). A moderate method is used in this research, where the metric used to measure the performance depends on the sum of the distances between the actual and proposed positions of each of the options in the slide bars in Figure 5.6.

Firstly, for the sake of obtaining a single numerical measure, the separate bands 'incorrect' and 'correct' are joined with a third band 'not sure' and considered as a single continuous spectrum. As in Figure 5.7, this single spectrum is now divided into seven regions; strongly incorrect, moderately incorrect, weakly incorrect, not-sure, weakly correct, moderately correct, and strongly incorrect.

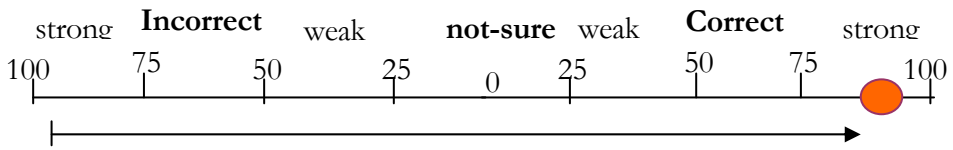


Figure 5.7 The notion of distance used in CBM test

The very weak approval or very weak denial is considered as 'not sure'. More levels may give more accuracy. There will be 7^4 possible combinations, though some are not practical (for example, all marked as strongly incorrect). Figure 5.7 illustrates the notion of distance (say d) when a incorrect answer is marked as strongly correct (here it is 6). On

the other hand, if a correct answer is marked as strongly incorrect, the distance will be 18 (3*6). The distance (d) may vary from 0 to 18. To get a positive metric for performance (say D) on a test item, all the distances for each answer are added and the total is subtracted from 36 (6*6). This metric varies from 0 for extremely erroneous belief, to 36 for exactly correct answer. In a real sense, the metric varies actually from 18 (selecting all 'not sure' or all incorrect or all correct gives 18) to 36. Therefore, any mark below 18 will be considered as 18.

CBM Method: Pros and Cons

Gardner-Medwin notes, "[Learners are] *more cautious in their expression of high confidence in exams than when doing formative assessment to aid study. …excessive diffidence or unwarranted confidence might disadvantage a student"* (Gardner-Medwin 2005, p. 1) The CBM method designed above will be used for formative assessment during the scaffolding process; and therefore, it may be expected, as Gardner-Medwin observed, that the students may not be extra cautious in expressing their true confidence level.

The complex testing methods such as the above may confuse learners. The following remarks of three students taken from Bush's questionnaire about 'liberal testing' methods reveal this fact; "*You are distracted by thinking about the best tactics for getting a high mark"*, "*It takes away your confidence*", and "*I was scared to answer a question*" (Bush 2001, pp. 161-2). However, Gardner-Medwin's observation is encouraging, "*With some students the principles involved have seemed complicated when explained, but present no problem once they gain experience*" (Gardner-Medwin 1995, p. 85). Moreover, there may be some concern about whether personality type or gender differences affect CBM methods. Gardner-Medwin denies this, *"Our data shows no evidence at all"* (Gardner-Medwin 2005, p. 7)

The proposed method provides elegant facilities for the learner to express their state of knowledge. This feature yields many advantages. Firstly, the system can precisely identify the misconceptions, and therefore, it can provide rich and relevant feedback. Secondly, since the uncertainty related to the learner's knowledge level is considerably reduced, the system could select a suitable scaffolding level (curriculum sequencing).

Moreover, this type of answering method requires significant levels of reflection (Gardner-Medwin 1995) and self-regulation. This, in turn, develops a high level of meta-cognitive abilities in the learners.

Alternatively, the learner may be allowed to select their preferred method of test schema among a set of limited options. However, they should know the testing method and its implications well before attempting the actual test. It is expected that the proposed method, compared to other similar methods, identifies the misconceptions and partial knowledge efficiently. However, it needs to be evaluated from a psychometric perspective (e.g. reliability, validity).

5.8 Mentor Model

The Mentor model plays a vital role in traditional ITSs. From an extreme point of view, Mentor model may be considered as an expert system for teaching that consists of the sufficient knowledge and skills of a typical teacher. In practical systems, Mentor models suggest or select, based on the information provided by Learner models, the appropriate pedagogical actions such as feedback, or next lesson material (example, revision material or next section). The selection process more or less resembles how a human tutor in such situations would do. Feedback should be adequate, optimal, and should be presented in a pedagogically pleasing manner. The next lesson material must be challenging as well as comprehensible to the student in their current state. As constructivist learning theories dominate educational system design, the role of the tutor in the model is changed from controller to mentor.

5.8.1 Pedagogical Action Selection

The term Pedagogical Action Selection (PAS) (Mayo 2001), as discussed in Chapter 3, refers to the acts of both selecting the remedy, usually feedback after a misconception is identified, and selecting the next pedagogical action. In this research, depending on the performance on formative assessments at the end of a scaffolding stage, the system should provide suitable pedagogical actions tailored to the learners. These actions may be, for example, providing suitable feedback and deciding the best next action such as selecting the next scaffolding level, or giving the same assessment or different assessment at the

same level. Different levels of feedback are included in the domain model for each misconception. Selecting the appropriate degree of detail and timing in the feedback process is important for the success of scaffolding in this research. Mason *et al* state: *"Feedback helps learners determine performance expectations, judge their level of understanding, and become aware of misconceptions. It also provides clues about best approaches for correcting mistakes and improving performance"* (Mason *et al.* 1999, p. 2). Moreover, the sub-concepts and the related series of mental states determine the different levels of scaffolding stages for a topic. Selecting the next optimal scaffolding stage for a learner is the other key task of the Mentor model. Depending on the system's estimate about a learner's ability, the next stage may be the same as the previous, at lower level, or at a higher level.

5.8.2 PAS Tables for Traditional MC tests

A table 2.1 is included in Chapter 2 that describes different levels of feedback suitable in different situations. Mason *et al* (1999) describe a decision tree (see Figure 2.3) for selecting a suitable feedback level depending on five factors. In this research, a number of tables (they are called PAS tables in this dissertation) are proposed to organize the potential pedagogical actions in different levels to suit different situations (e.g. correct or incorrect on the recent MC test). The cell entries include not only feedback levels but also curriculum sequencing options. Experienced human tutors take different (levels of) pedagogical actions for different types of learners, based on their current performance (e.g. correct or incorrect on the recent MC test), general abilities, traits and other mental qualities. The PAS tables reflect experienced teachers' decision-making processes in different situations (see Tables 5.1 & 5.2)

Levels	PASc- for Correct answers
L1	Affirm (just inform that the answer is correct) Move two levels forward
L2	Explain why the selected answer is correct Move to next level
L3	Explain why the selected answer is correct & why other answers are wrong, Move to next level
L4	Explain why the selected answer is correct & why other answers are wrong Provide topic related explanation. Move to next level
L5	Explain why the selected answer is correct & why other answers are wrong, Provide topic related explanation Stay at same level

Table 5.1 Pedagogical Actions for Correct Answers

First, consider the case where MC tests are used as the scaffolding blocks. In this case, the performance on a test has binary values: correct or incorrect. As stated before, feedback levels and curriculum sequencing options are combined, and five levels of pedagogical actions (or simply PAS categories) are created for both cases: incorrect answers (PASwL1 to PASwL5) and correct answers (PAScL1 to PAScL5). Tables 5.1 and 5.2 describe these PAS categories in detail. The feedback contents related to the PAS categories in both cases increase with their levels. The higher level PAS category is designed to address the neediest students. For example, the research in feedback reveals that when a learner has high confidence in their selected answer, and if the answer is also correct, they just need a verification (Kulhavy 1977). Usually, the talented students will have high confidence in their answers (feedbacks related to confidence levels will be addressed in next section). The level-1 PAS for correct answer (in Table 5.1, denoted by PAScL1) describes this type of feedback. In this category, the learner is also considered to be capable of skipping the next scaffolding level.

Levels	PASw - for Incorrect answers
L1	Let students Answer-Once-Again (same MCQ) If answered correct second time, give MCQ at the next Scaffolding Level (move to one level up); otherwise, move to L3
L2	Explain why the selected answer is wrong, and let students Answer-Once-Again (same MCQ). If answered correct second time give MCQ at the same Scaffolding Level (stay at same level); otherwise, move to L5
L3	Explain why the selected answer is wrong & why the system's answer is correct , Give another MCQ, but at the same Scaffolding Level
L4	Explain why the selected answer is wrong & why the system's answer is correct Compare the concept with related concepts in Z and UML, if applicable Give another MCQ, but at the same Scaffolding Level
L5	Explain why the selected answer is wrong & why the system's answer is correct , Provide topic related explanation. Compare with Z and UML, if applicable. Give another MCQ, but at the next lowest Scaffolding level (move to one level down)

Table 5.2 Pedagogical Actions for Incorrect Answers

The feedback type PAScL5 (Table 5.1) indicates that the system assumes the given MC test is very hard for the particular learner, and therefore they need some informative feedback even if they have answered it correctly (it may be a lucky guess). However, subsequently, the learner will be given a problem at the same scaffolding level to enable them to prove that the previous success is not merely a lucky guess. Whereas, the level-5 PAS for wrong answer (PASwL5 in Table 5.2) indicates that, in addition to giving detailed feedback, the learner will be encouraged to return to the previous scaffolding

level. A simple system may select a suitable PAS options for a certain situation based on the table look-up (see Chapter 6), or a fully learner-controlled system may just suggest a suitable option while allowing the learner to select any other option (see next section). Chapter 6 discusses a strategy to select a PAS category from these tables in uncertain situations.

Basically, all the feedback in this design is immediate. Delaying feedback may be useful in some cases when deliberately guiding the learners to enter deadlocks in order to give them an opportunity to figure out their own misconceptions. Kulik *et al*, in their seminal paper 'Timing of Feedback", conclude *"the delayed feedback appears to help learning only in special experimental situations and that more typically to delay feedback is to hinder learning"* (Kulik *et al.* 1988, p. 95). Being part of the scaffolding process, the immediate feedback will be expected to be effective in this research.

PAS Tables for Confidence-Based MC tests

So far the traditional MC tests have been considered for scaffolding. As discussed earlier, formative assessment plays a major role in the scaffolding process; and Confidence-Based MC tests are very efficient for formative assessment. To accommodate Confidence-Based MC tests in this design, there are many options; the performance, which can be denoted by P, may be considered binary as before, or it may be divided into different distinct ranges. If it is left as binary, there are two options: it may be considered as 'not fuzzy' with two concrete states (e.g. correct (34-36) and incorrect (18-34)) or 'fuzzy' with two overlapping sets (for example, correct (33-36) and incorrect (0-33), where P=33 means the performance is 50% correct and 50% incorrect. P may assume the fuzzy membership mappings given in Figure 5.8a. However, naturally, in confidence based MC tests, the performance cannot be defined concretely, and therefore, P will be treated as fuzzy in this research for confidence-based MC tests. Tables 5.1 and 5.2 may be used for curriculum sequencing and feedback. However, it is now possible to provide richer feedback based on the learner's belief levels on each answer option. For example, if a learner shows high confidence in a misconception, they will naturally be more anxious to know why their answer is wrong. As Kulhavy *et al* put it, "… [in this situation], *students are likely to spend a substantial amount of time in trying to locate the source of their mistakes…."*

(Kulhavy 1977, p. 225). The appropriate feedback level may be selected to suit different levels of misconceptions related to the relevant answer options.

Alternatively, the performance variable P may be more precisely defined. For example, P may take three values: incorrect, medium, and correct. Further, it may be fuzzy or not; but only the fuzzy case will be considered in the following discussion. Figure 5.8b shows a fuzzy membership mapping for P with the following ranges: incorrect (0-33), medium (33), and correct (33-36). Note that the ranges are overlapping. If the measurement is continuous, the mapping will also be continuous (i.e. incorrect (0-34), medium (32-34) and correct (32-36)).

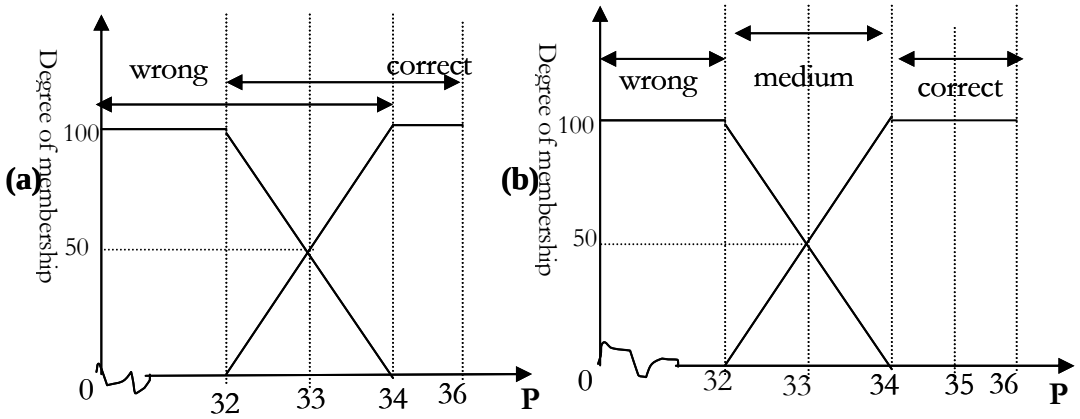


Figure 5.8 Fuzzy Membership Functions for Performance (P)

In a CBM scheme, even for an MC test, in addition to correct and incorrect, a 'medium' level performance can also be identified. It is essential to design a suitable PAS table for each case. Tables 5.1 and 5.2 can be used again for 'correct' and 'incorrect' with substantial modifications. Table 5.3 gives the proposed PAS levels for the new performance category 'medium'. However, as stated before, since the learner's level of misconception on each answer option is available, it is now possible to provide richer feedback based on individual misconception. For example, the 'answer once again' option may be irrelevant in confidence-based testing as the degree of slip (or guessing) is drastically reduced (unless only one answer is selected as 100% correct, when it is incorrect).

The perceptual state of the learner, as Kulhavy *et al* (1989) note, influences the effect of feedback on the learner more than the content of the feedback. In particular, the initial confidence level significantly influences the effectiveness of feedback (Kulhavy 1977;

Kulhavy *et al.* 1985; Kulhavy *et al.* 1989; Kulhavy *et al.* 1990). Feedback should be carefully designed to go well with different confidence levels.

As mentioned before, the main instructional strategy used for mentoring in this study is scaffolding. Designing instructional scaffolding involves two steps: designing scaffolding blocks and designing an instructional strategy to realize the scaffolding process. The first step was completed in the previous section (Domain model). The next section will describe how the process of scaffolding abstraction could be implemented in a fully learner-controlled interactive learning environment.

Level	PASm - for Medium level answers
L1	Explain why the answer is correct & why some other answers (for which the learner indicated high confident for correctness) are wrong, Move to next level
L2	Explain why the selected answer is correct & why all the other answers are wrong Move to next level
L3	Explain why the selected answer is correct & why all the other answers are wrong, Provide topic related explanation, Move to next level
L4	Explain why the system's answer is correct and other answers are wrong Provide topic related explanation. Give MCQ at the same Scaffolding Level
L5	Explain why the system's answer is correct and other answers are wrong Provide topic related explanation. Compare with Z and UML, if applicable Give MCQ at the same Scaffolding Level

Table 5.3 Pedagogical Actions for Medium Answers

5.8.3 Fully Learner-Controlled Environment

An interactive learning environment should provide a suitable environment for a learner to build the intended knowledge and skills on their own initiative. As stated before, after the modelling phase in LOZ, the system assumes that the learner has acquired sufficient mental strength to perform a task related to the basic mental state. This assumption is assessed using relevant formative tests. The learner may simply answer the question. Alternatively, they may ask for a worked example or revisit the previous lesson. If they opt to answer, they will be immediately notified of the result. If it is wrong, they may try again. In both cases, they may choose the feedback level they prefer. The system allows the learner to select his/her own feedback and next topic. Allowing the learners to choose their levels of feedback is not a new concept. While discussing the provision of increased learner control over the type and elaboration of feedback, Mason *et al* states, "*This option*

avoids the presentation of unnecessary feedback, yet allows students to receive more clarification as desired" (Mason *et al.* 1999). Moreover, the scaffolding levels are designed to suit potentially weak students. The learner can select a suitable scaffolding level at any time.

Enriched by the constructive learning theory, the environment described above is totally controlled by the learner. This type of CBL system was highly promoted by some researchers in the aftermath of so called failures of Learner models. Designing complex ITSs is considered wasteful and CBL systems are seen as merely cognitive tools (Salomon 1993; Jonassen 1995), which learners could manipulate to construct their own knowledge. Shute *et al* (1995) describe the history of this trend under the heading "1990s: Great Debates" (p. 583). Implementing a fully learner-controlled system is not as easy as one may perceive. This type of CBL system may be beneficial for a highly cultured learner who can effectively control their learning process. It demands a higher level of meta-cognitive abilities such as reflection and self-regulation. Nevertheless, a question debated in CBL circles is whether the learners are ready to get benefits from this type of instructional strategy, or in other words, how much learner control is advantageous for learners? In an attempt to argue for the usefulness of a Learner model for the proposed system, some experiences of the leading researchers in this field will now be considered.

The benefits of using CBL systems for learning depend on various factors such as the domain type, learning-outcome and learner traits. For example, an exploratory learning environment is better if the learning-outcome is building a 'functional mental model', and for learners who are systematic and high-explorative. Shute *et al* conclude "… *a midpoint between too much and too little learner control is probably the best bet as far as optimal ITS learning environment. Furthermore, it should not be fixed, but flexibly changing in response to the learner's evolving needs*" (Shute *et al.* 1995, p. 584). While commenting on the expectation that educating students may be possible by just giving access to the information (especially the internet), Laurillard states, "*The notion is often accompanied by the rhetoric of being student-oriented, or learner-oriented: it is as absurd to try and solve the problem of education by giving people access to information as it would be to solve the housing problem by giving people access to bricks. Part of the point of an education is to give people the skills and understanding to enable them to handle information"* (Laurillard 1996, P. 6). In an attempt to argue for Learner Models while

compromising with constructive theory, Self states, *"In general, there is a set of potential properties available, each of which is more or less desirable than others, and there needs to be some heuristic strategy for guiding the students towards events which, on balance, are more likely to lead the learner to encounter contexts which allow a continuing learning experience"* (Self 1999, p. 360).

The desired outcome of using the proposed system in this research is a smoothly executable skill rather than building a functional mental model. To achieve this goal, as discussed before, an exploratory learning environment is not very suitable. Therefore, based on the foregoing arguments, it was decided that LOZ should accommodate some sort of Learner model which could assist the learners (at least while they learn through the scaffolding process). In the next chapter, the proposed design for the Learner model that provides limited adaptability based on locally intelligent artefacts will be discussed.

5.9 Refinement Unit

As argued before, a Refinement unit that automatically transforms Object-Z specifications to equivalent Java code may be incorporated in the system. This feature could extrinsically motivate learners. Additionally, the system may provide an online environment which leads learners through a series of refinements in order to get the final program code for their own specifications. This facility, if provided, would enable the learner to engage in a constructive learning process based on problem transformation with scaffolding. Both ZAL/ZED and MEMO-II use animation for teaching (Section 3.3). ZAL/ZED transforms Z to LISP, and gives a limited interactive environment for 'try and see' learning. MEMO-II transforms an algebraic formal language automatically to some high level languages, and is intended for learning programming. There is no full-fledged animation system for Object-Z (McComb *et al.* 2003).

Animation of formal specification is an important research discipline on its own. Moreover, providing an online environment for animating Object-Z notation requires much effort. Nevertheless, in phase-I of the design process, only multiple choice tests are used for assessments and free form questions are avoided. A totally new specification will not ever be created while learning in phase-I, and therefore, automatic translation is not required. The Refinement unit in phase-I may include a mapping from the existing

specifications (related to MC tests and examples) to a collection of Java programs, and an efficient searching algorithm.

5.10 Summary

An Instructional strategy for CBL systems based on problem transformation and scaffolding has been devised. The key aim of this strategy design is to alleviate the potential difficulties associated with learning complex subjects such as Object-Z notation. The system has two phases, and this chapter has discussed a design for the first phase. The second phase will be discussed in Chapter 9.

A survey was conducted to evaluate the effectiveness of using a popular formal method tool for learning a formal notation (Z). The survey shows that the tool is not suitable for learning Z (but, it is useful for its original tasks such as specification creation and type-checking). Thereafter, the key challenges specific for learning Object-Z, the exemplar domain in this research, were identified. Creating a formal Object-Z specification from scratch (based on the given informal textual description only) is a challenging task as it demands both OO abstraction and mathematical abilities. On the other hand, transforming a semi-formal UML model to an Object-Z model is comparatively easy, because a UML model is basically a form of abstraction of the original problem. Therefore, the transformation process requires mathematical ability only. Initially, learners are required to construct Object-Z models from the given UML models. Gradually, the UML support is withdrawn. A four-phase instruction model is used to realize the scaffolding process. The pre-condition phase is included to revise the necessary UML knowledge. A Refinement unit, that can also facilitate active learning through problem transformation, is proposed to keep extrinsic motivation high.

The architecture of the CBL system has been designed to facilitate the proposed instructional strategy. The domain model plays an important role in our design. Concepts to be learned are organized in a hierarchical structure. The notion of mental states is discussed. While learning a concept progressively, students go through a series of mental states relevant to that concept. Each mental state is considered as a scaffolding level. MC tests are used for formative assessments. All the other course materials such as MC test

items, distracters related to misconceptions, and the corresponding feedback material have been carefully designed to support the scaffolding process.

In the Mentor model, various options for pedagogical actions (combining feedback and curriculum sequencing), which reflect typical teachers' actions in different situations, are organized in different tables. The options are ranked: the highest level option is designed for most needy students, whereas the lowest aims at gifted students. Based on the performance of formative assessment, the system could lead a student from one scaffolding level to another. Traditional MC tests are not capable of measuring performance accurately. There is no way to identify partial knowledge on a test item. Therefore, a Confidence-Based MC test schema has been designed, where more than two levels of performance measurements are possible (for example, correct, medium, and incorrect).

There are many ways to implement scaffolding. A fully learner-controlled system may let the learner decide everything: topics to be learned, scaffolding steps, and feedback level. This environment demands significant commitment from the learner and is not suitable for everyone. A Learner model, which provides appropriate assistance to individual learners, is desirable. The next chapter describes the Learner model of LOZ in detail.

Chapter 6

Designing a Fuzzy Learner Model

6.1 Introduction

The overall architecture of LOZ, and in particular, the Domain model and Mentor model that formulate a fully learner controlled CBL system, were discussed in the previous chapter. In this chapter, a fuzzy logic based Learner model will be designed progressively. Firstly, the notion of a locally-intelligent Learner model is discussed. After that, a simple Learner model is illustrated (Section 6.3). Later, this model is used as a basis for designing the proposed fuzzy Learner model. The new enhanced Learner model now uses possibility theory to handle uncertainty. However, probability theory is also used to complement the model. Section 6.4 discusses the features necessary to predict the relevant pedagogical actions under uncertainty. In this model, the different levels of PAS options, defined in Chapter 5, are considered fuzzy variables. A set of fuzzy rules, which resemble experienced teachers' decision making processes in typical learning situations, is included in the model. The rules are used to select an appropriate level of PAS option for a learner after a formative assessment based on his/her performance.

Section 6.5 discusses the dynamic nature of the proposed Learner model. As time goes by, learners may forget or learn from other sources. Another set of fuzzy rules are included, in order to handle the dynamic feature. A time threshold is suggested to prevent senseless predictions. To update the Learner model based on new evidence, Bayesian Network (BN) theory is used. The updating process, including fuzzy to BN transition, and vice versa, is discussed in Section 6.6. The process of bookkeeping for the Learner model is discussed in Sections 6.7 and 6.8. Externalising Learner models is found to be effective as learner may provide their own estimation if necessary. Section 6.9 discusses issues related to opening the proposed Learner model (and Mentor model) to the students, peers and mentors.

6.2 Learner Model Complexity

Basically, the Learner model consists of a collection of the system's beliefs about a learner (Holt *et al.* 1993). The learner may explicitly provide some detail, and/or the system may infer some information, based on the learner's interactions. The Learner model is essential for providing motivational and adaptable learning environments tailored to a variety of learners (Greer *et al.* 1994). However, for a long time, designing and implementing optimal Learner models have been considered a challenging task in CBL research. The seminal paper by Self (1988), described it as *"the intractable problem of Learner modelling"*, and suggested several possible remedies. Learner models were discussed in Chapter 3.

For decades, a basic question has been asked and discussed within the CBL community: does a CBL system need to be adaptable, and if so, how far should it be adaptable (Greer *et al.* 1994)? To achieve adaptability, it is obvious that the learners need to be modelled. Therefore, the question must be asked: is it necessary for a Learner model to be optimal? The research described in this dissertation, similar to the classical research of Katz *et al* (1994), assumes that the Learner models need not to be optimal. Katz *et al* state, *"We share with Derry et al (1992) the belief that in low-risk decision making situations such as tutoring, where new information is constantly made available for modifying diagnostic hypotheses, imprecise Learner modelling is adequate"* (Katz *et al.* 1992, p.102). Learners are capable of adapting to a variety of teachers who in turn make sub-optimal assumptions about students. The Learner model in LOZ is not based on rigorous mathematics to minutely differentiate learners' abilities, mental states, preferences, etc. Instead, it will generally avoid the type of extreme system behaviour which may lead the learner to doubt the usefulness of the system for learning (for example, giving feedback about a careless mistake to a student who has already mastered the topic).

6.2.1 Locally-intelligent Learner Model

The Learner model designed in this research is inspired by the notion of a 'bi-modus learning environment' (Otsuki 1993), where the concepts of ITS and ILE are merged to include the advantages of both. The proposed model will enable the system to provide an appropriate environment for learners to build up their own understanding on the subject,

and to develop their meta-cognitive abilities whilst also providing relevant guidance tailored to individual users. Globally, consistent Learner models are expensive to create and they are not essential for low-risk tasks such as teaching: this research incorporates locally- intelligent constructs. McArther *et al*. describe the Learner models of this type as, *"Rather, they are locally-intelligent agents that impose islands of tutor control in a relatively large expanse of student-controlled inquiry activities"* (McArthur *et al.* 1999, p.24). Learners will not be interrupted unless they ask for help or are in an adverse situation where the system senses that they cannot proceed any further without help. Even so, the learner may discard the advice offered.

Locally-intelligent systems may not model the learners' knowledge state accurately. Less precise Learner models are usually sufficient. As McArther *et al* state, *"… it is not critical that the local agents have a complete or fully accurate cognitive model of the student's misconception"* (McArthur *et al.* 1999, p.24). Even the best teachers do not make pedagogical decisions based on the global learning history of individual students but, instead, they usually act on a short learning history and immediately available evidence. However, in general, this evidence is rich. As Holt *et al* state, *"Human teachers have a wider variety of cues to work with than computerized learning systems. A "eureka" look, a puzzled expression, or a hesitant tone of voice may all shape remedial action"* (Holt *et al.* 1993, p.26). In the context of handling uncertainty in Learner models (which will be discussed later in this chapter), the locally-intelligent models assume a limited dependency between knowledge elements, and therefore, they will include a collection of simple causal networks instead of a single complex network. This feature reduces the complexity of the Learner model, and in turn, significantly increases the efficiency. The Learner model in LOZ will be locally-intelligent and, therefore, it should be simple but reasonably efficient.

6.3 A Simple Learner Model

Various types of Learner models and their features were discussed in Chapter 3. Firstly in this section, a very simple Learner model that provides a relevant scaffolding process and suitable feedback will be discussed, and progressively, the design for a locally-intelligent Learner model will be described. As outlined in Section 5.6, the domain knowledge in LOZ is organised in a tree format (see also Figures 5.2 & 5.4). The important entities in

the tree are the 'main-concepts'. Every 'main-concept' is associated with a pre-condition concept, a series of sub-concepts, and a set of exploration concepts. A sub-concept is associated with a sequence of mental states (from a basic mental state to a target mental state – see Figure 6.1). Each mental state is considered as a scaffolding stage.

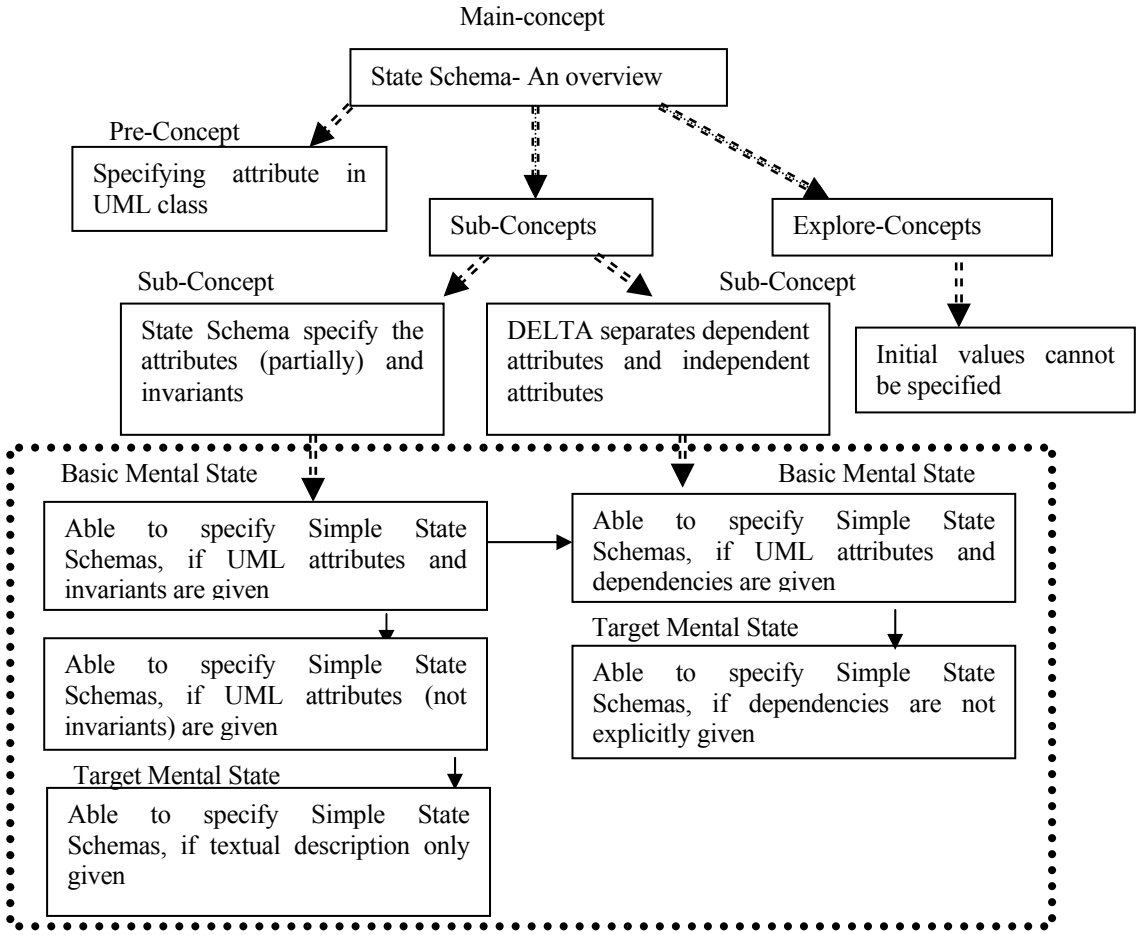


Figure 6.1 Lesson Organization (including Mental States)

After learning the pre-condition concept and the first sub-concept, the system expects that the learner will be in the basic mental state associated with the first sub-concept. At this juncture, the system will check the learner's strength in the basic mental state using formative assessments. If it is at the desired level, the system will lead the learner to the next suitable scaffolding stage. As mentioned previously, the MC tests are used for formative assessments in order to measure the learner's strength in the current mental state. The distracters in the MC tests are designed in a manner so that any possible misconceptions can be identified[5]; and therefore, they may be easily isolated and treated.

[5] Confidence-based MC tests can identify more precisely the potential misconceptions.

A simple system, without a Learner model, may use the performance of a student on a given test item to select the feedback and the next scaffolding stage. However, these pedagogical decisions may not be appropriate: for example, the given test may not be suitable. A single performance measure is not sufficient to gauge the knowledge level of a learner. Therefore, in order to provide adaptable support, the individual learner's state of knowledge (or strength of mental state) should be modelled. It should be noted that the knowledge level for a learner is dynamic and therefore any such model will also be dynamic.

Strength of Mental State

A numerical variable (ranging from zero to 100) known as a "Strength of Mental State" (SMS) is used in this research, in order to measure the learner's level of potency in a certain mental state. SMS represents the increasing strength in a particular mental state as it moves from zero to 100. Naturally, for any particular learner, the current value for SMS will depend on their previous SMS values. Basically, SMS may be associated with the traditional 'overlay' concept (Holt *et al.* 1993). Notwithstanding, in this research, the semantics of this measure are interpreted through the constructivist approach. As previously mentioned, the Learner model in LOZ will be locally-intelligent, which means that the SMS related to a main-concept will not have any impact on the SMSs related to the other main-concepts. However, the SMSs related to the same main-concept may impact on each other in a limited manner (see Figure 6.1).

SMS may be interpreted in quantitative terms or in probability terms. In a quantitative interpretation, the value SMS=50 means that the system believes that the learner is 50% strong in the corresponding mental state and alternatively, in probability terms, it can be interpreted as the system believing 50% that the learner is 'sufficiently strong' in the basic mental state. An initial value for SMS_0 should be assigned. For convenience, it can be assumed that 50 will be the initial value for basic mental states of all the first sub-concepts. Which means that, after learning the materials relevant to the pre-condition lesson and the first sub-concept, the strength of the basic mental state will be 50.

After answering an MC test, the system has then acquired some evidence about the learner's strength in the current mental state. The following heuristic formula is used in

this research to update SMS depending on the performance. P denotes the variable performance, which can assume two values – Correct or Incorrect. The resultant SMS values are bounded by zero and 100.

$$\text{If P is Correct,} \quad SMS_{i+1} = SMS_i + (100 - SMS_i) * \frac{J}{N+1}$$

$$\text{If P is Wrong,} \quad SMS_{i+1} = SMS_i - SMS_i * \frac{J}{N+1}$$

Where, N is the total number of scaffolding steps associated with a sub-concept,
J is the number of remaining steps to be completed,
SMS_i denotes the SMS value of the current mental state after the i^{th} MC test.

The above formulae are based on the notion of a traditional 'point-scoring' scheme (Katz *et al.* 1994). A weak student will be awarded higher points than a stronger student if they both perform equally well on a test. It may be a lucky guess in an MC test, but that possibility is not considered here. Similarly, a strong student will be given a larger penalty than a weak student if they both perform equally badly. This may be a slip-up, but that possibility is also not considered here. Moreover, more marks will be awarded (or deducted) in the earlier rather than in the later stages of the scaffolding process.

There are several heuristics used in this research. Stern *et al* discuss this issue in relation to their MANIC system: *"Many of the decisions made in MANIC are rather ad hoc, with numbers determined by domain experts and instructors. However, this is not the fault of MANIC alone: many systems rely on a priori formulas designed by instructors. But, we are not satisfied with this as a solution"* (Stern *et al.* 1998, p.581). Heuristics are inevitable in life. For example, we used to set pass marks for exams with 50% and above being a pass. The difference of knowledge between those students who were on either side of 50% may have been negligible, whilst the impact on their results would be significantly different. Stern *et al* (1998) discuss some machine learning techniques to improve the heuristics used in MANIC. The heuristic formulae used in this research can also be tuned, using empirical methods or machine learning techniques (Kiat 2003).

After updating the SMS, the system will attempt to provide suitable feedback and curriculum sequencing. Different levels of PAS options for different performance levels have already been designated in Chapter 5. A higher level PAS option is intended for a

lower level student. For any particular learner, the current SMS can be used to determine the appropriate level of PAS option in the present situation. For this purpose, SMS is divided into 5 ranges: 0–20, 21-40, 41-60, 61-80, 81-100. Table 6.1 gives the SMS ranges and their corresponding PAS levels for the performance categories Correct and Incorrect.

SMS- ranges	P - Incorrect	P- Correct
81-100	PASwL1	*PAScL1*
61-80	PASwL2	*PAScL2*
41-60	PASwL3	*PAScL3*
21-40	PASwL4	*PAScL4*
0-20	PASwL5	*PAScL5*

Table 6.1 SMS and Pedagogical Actions

Depending on the range of SMS, the required level of PAS option will be selected. SMS in the lower range indicates that the corresponding learner needs a higher level PAS option. For example, if a learner with a low SMS answers a test incorrectly, detailed feedback will be given and, subsequently, the learner may also need to take another test at the same scaffolding level.

The mental states, in the series associated with the same sub-concepts, are sequentially dependent (Figure 6.1), and therefore, the following formulae are used to initialise the SMS related to the subsequent mental states in the series:

$$\text{If } SMS_{last}^{\,j} > 50 \;, \quad SMS_{first}^{\,j+1} = 50 + (SMS_{last}^{\,j} - 50) * \frac{1}{N+1}$$

$$\text{Otherwise} \;, \quad SMS_{first}^{\,j+1} = 50$$

Where, N is the total number of scaffolding steps associated with a sub-concept, $SMS_{i}^{\,j}$ denotes the SMS value of the j^{th} mental state after the i^{th} MC test of the current sub-concept.

As previously explained, before moving to the subsequent mental state (if the SMS of the previous mental state in the series is less than the current SMS) all the previous SMSs in the series, that are less than the current SMS, will be updated by the current SMS, which means that the corresponding SMSs of the mental states in a particular series will always

be in descending order. For example, if a learner's strength is 80% in specifying a state diagram without UML support, they cannot have less than 80% strength in doing so with additional UML support. Within the same main-concept, only the basic mental states of different sub-concepts are considered dependent (Figure 6.1). The following formulae are used to initialise the strength of the basic mental state associated with the subsequent sub-concepts of the same main-concept.

If SMS_{final}^{1} (sub-concept $_i$) >50 ,

$$SMS_{first}^{1}\text{ (sub-concept }_{i+1}) = 50 + (SMS_{final}^{1}\text{ (sub-concept }_i) - 50) * \frac{1}{N+1}$$

Otherwise,

$$SMS_{first}^{1}\text{ (sub-concept }_{i+1}) = 50$$

Where, N is the number of sub-concepts associated with the current main-concept, $SMS_i^j(\text{sub-concept}_k)$ denotes the SMS value of the j^{th} mental state after the i^{th} MC test of the k^{th} sub-concept of the current main-concept.

Nevertheless, designing and maintaining even a locally-intelligent Learner model is challenging, primarily due to the uncertainty associated with the sources of evidence concerning the learner's knowledge state. The above described method is not efficient. Although, different levels of PAS options are designed to deal with uncertainty to a certain extent, only a single table-look-up is used as the selection mechanism in this method (Table 6.1). Moreover, SMS values are updated using a simple point-scoring scheme. Therefore, based on the above model, a more efficient strategy is designed to handle uncertainty in the Learner model. This strategy will be described in detail in the next section.

6.4 Determining Pedagogical Actions under Uncertainty

A CBL system equipped with an impaired Learner model due to uncertainty could provide inappropriate feedback or repeatedly offer an untimely sequence of lessons to learners. As a result, learners may become confused or irritated, and ultimately, their motivation could be affected. In Chapter 3, various approaches to handling uncertainty in Learner models were discussed. Some of the issues, pertinent to the proposed Learner model design, will be outlined in the next section.

6.4.1 Uncertainty Handling

For complex, real-world domains, probabilistic inference using BN is computationally expensive (Cooper 1990). Villano states "*the belief propagation in a* [Bayesian Belief Network] *is a complicated issue"* (Villano 1992, p.496). However, there are many efficient algorithms available for some special classes of belief networks (Pearl 1988). BN is widely used for uncertainty handling in CBL systems. Systems such as DT Tutor (Murray *et al.* 2000) and ANDES (Conati *et al.* 2002) use various algorithms for belief propagation in their Learner models.

In the ACT Programming Language Tutor, the knowledge nodes (programming rules) are assumed to be totally independent (Corbett *et al.* 1992). In this research, the proposed Learner model assumes only limited dependency between the knowledge nodes. The mental states related to the different main-concepts are considered independent. In general, the propagation overhead in the 'locally-intelligent' systems will be negligible, since their Learner model uses a collection of simple belief networks, rather than a single complex network. The belief propagation is efficient in this model since the networks are not complex, because they are limited to the mental-states related to a particular main-concept.

In the 1990s, Hawkes *et al* (1990) first proposed a design for a Learner model based on fuzzy set theory. Later, Katz *et al* (1994) used possibility theory (Hopgood, 2000), the outgrowth of fuzzy set theory, in SHERLOCK-II to handle uncertainty associated with diagnosis. They use the term 'knowledge state' for a possible level of trainee competence in a skill or concept. In order to apply fuzzy rules, they use a type of point-scoring scheme to update the strength of knowledge states. Fuzzy logic is more natural for human understanding, than mathematical models such as BN. In particular, for tasks such as teaching and learning, where vagueness is inherent, fuzzy logic based decision making (and reasoning) will be more suitable than purely numerical approaches. In this research, possibility theory is used to handle uncertainty in the Learner model (basically for prediction). However, BN theory is also used for updating the belief network. The rationale behind this decision will be discussed later in this chapter.

After using formal models for diagnosis, various heuristic and decision theoretic approaches are used for curriculum sequencing. As discussed in Chapter 3, systems such as CAPIT and DT tutor employ decision theoretic approaches and use utility models for curriculum advancement. Unlike CAPIT, the curriculum advancement options (the next possible scaffolding stages) in LOZ are limited. Therefore, the utility values for the potential scaffolding stages (previous, current, next, etc.) can be incorporated with different levels of PAS options. For the sake of uncertainty handling, these levels of PAS options are considered as fuzzy variables. To identify the next suitable scaffolding stage and appropriate feedback pertinent to an individual learner, a number of PAS variables together with a set of suitable fuzzy rules are used in this research. For example, the level-1 PAS (Table 6.1) for the correct answer suggests that the learner will just receive confirmation that their answer is correct, and they may then skip the next scaffolding stage and move two steps forward. The next section will discuss the strategy used for PAS in LOZ to alleviate the effect of uncertainty.

6.4.2 Adaptable Scaffolding

The strategy used for PAS is critical in CBL systems, since there is a potential for the system's trustworthiness to become questionable and, eventually, the user may abandon the learning process. During the scaffolding process the system frequently measures the learner's performance. It could be expected that a learner with a strong mental state will answer an easy question correctly and, similarly, a learner with a weak mental state will answer a difficult question incorrectly. However, this is not a certainty, since a lucky guess or careless mistake may still play a role. A gifted learner may be annoyed if detailed feedback is given for just a careless mistake. Similarly, a weak learner may be frustrated if the system asks the learner to try again to solve a problem to which they have not a clue to the answer and it may well be beyond their Zone of Proximal Development.

To incorporate adaptability in the scaffolding process, as previously mentioned, the Learner model in LOZ retains some numerical measures, including the Strength of Mental States (SMS). Earlier, a simple method was described to determine the required level of PAS option, based on the variables P and SMS. The ranges of the variable SMS, specified in the look-up table (Table 6.1), are discrete. The boundaries do not overlap. In probability terms, the variable SMS is treated as a binary variable, which means SMS is

true if the learner's strength in the current mental state is 'sufficiently strong' to perform a certain task, since otherwise it would be false. However, this interpretation is not solid as the phrase 'sufficiently strong' is rather vague. Corbett *et al* (1993) keep a similar variable to aid knowledge tracing, which takes one of two discrete values: learned or un-learned.

Alternatively, instead of a single state 'sufficiently strong', different linguistic terms such as 'strong', 'medium' and 'weak' may be used for SMS to describe different potential states. A binary variable (true or false) may be associated with each state to denote the corresponding system's belief. In a similar approach, Mislevy *et al.* (1996) in HYDRIVE use linguistic terms, such as 'expert' and 'good' to represent different knowledge levels, and they use BN for Learner modelling. Though these states look like fuzzy ones, in actual fact they are not. According to their definition, an 'expert' cannot be 'good' at the same time. In contrast, during real life situations such as learning and mentoring, these clear-cut distinctions cannot be made for variables which may include the knowledge level of a learner and/or the difficulty level of a question (this will be discussed in detail later in this chapter).

Naturally, the variable SMS, which represent the potency of knowledge, cannot take concrete yes-or-no types of values and, therefore, should be treated as a fuzzy variable. Hawkes *et al* state, *"The use of fuzzy terms allows for imprecision and vagueness in the values. This provides flexible and realistic representation that easily captures the way in which the human tutor might evaluate a student"* (Hawkes *et al.* 1990, p. 416). In a simple case, the variable SMS may take two linguistic values with overlapping ranges: 'sufficient' (say 40-100) and 'not sufficient' (say 0-60). The strength of a student, with a SMS between the values 40 and 60, will be both 'sufficient' and 'not sufficient' to a certain degree. In this research, the SMS takes three linguistic values: strong, medium and weak. For a given SMS, the degree of membership for these sets will be determined by a membership function (Figure 6.2). For example, a score SMS=75 means that the system believes the learner is at the 75% strong and 25% medium levels, related to the current mental state. By convention, this will be denoted as $\mu_{Strong}(75) = 0.75$, $\mu_{Medium}(75) = 0.25$.

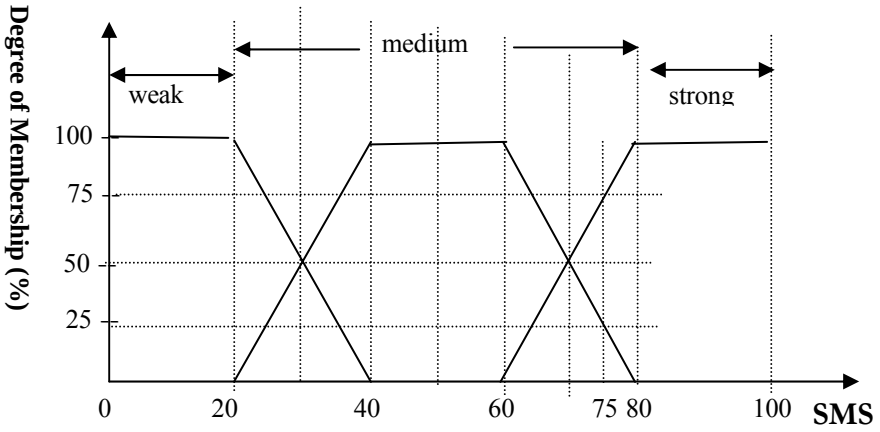


Figure 6.2 Fuzzy Membership Functions for SMS

In this research, unlike the other CBL systems that use fuzzy logic for Learner modelling, a definite membership functions is used. Katz *et al* (1994) use five overlapping sets to model the variable 'knowledge state' of a student. Hawkes *et al.* (1990) use seven intervals to model a variety of variables, from motivational state to error count. Each interval is assigned a number ranging from one to seven. Neither research group employs a single crisp value to represent the 'knowledge state'.

After an MC test is answered, the performance level P is measured. For a traditional MC test, P is just a binary variable and takes two concrete values: correct or incorrect. As previously discussed, if the confidence-based MC tests are used, P can be a fuzzy variable and may take three linguistic values: Correct, Incorrect, and Medium (Figure 5.8). In Section 5.3, different levels of PAS options were designed for each type of performance (Tables 5.1, 5.2, and 5.3). Similar to SMS, it is natural that the boundaries of different levels of PAS options cannot be defined strictly. Basically, the PAS levels are different linguistic values related to the possible pedagogical actions. Therefore, the PAS levels for each type of performance can be treated as a fuzzy variable. The PAS variables for Correct, Incorrect and Medium performances are named PASc, PASw and PASm, respectively. The different levels of the variable PASc will be, in their increasing levels, denoted by PAScL1 to PAScL5 (similarly, the notations PASwL1 to PASwL5 and PASmL1 to PASmL5 will be used for PASw and PASm, respectively). The fuzzy set membership function in Figure 6.3 is given for the variable PASc. The same function will also be used for the other two PAS variables. For example, PASc=87.5 means the required level of PAS option is both, PAScL5 and PAScL4 at 50%.

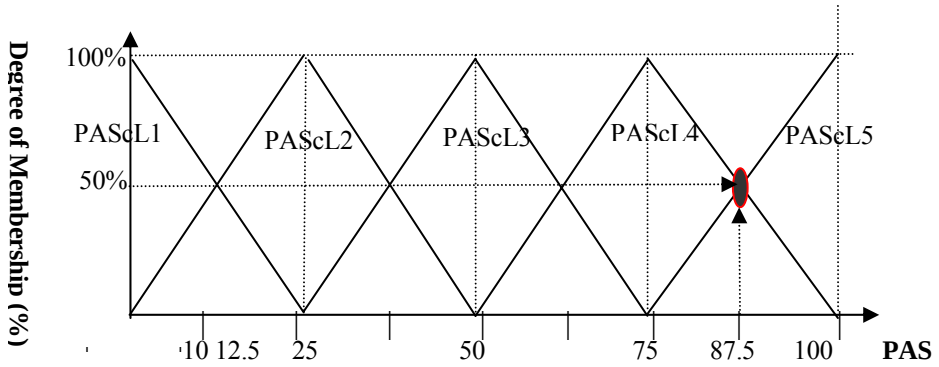


Figure 6.3 Fuzzy Membership Functions for PAS

Furthermore, in the previous technique, PAS levels are selected based on the variables SMS and P only (Table 6.1). In addition to SMS and P, the difficulty level of the question also plays an important role in determining suitable pedagogical requirements. The domain expert may initially propose the difficulty level for each question. Naturally, the difficulty level for an MC test will closely correlate with the related scaffolding stage. Some questions are considered difficult and some others are considered easy. However, this division is usually vague − clear-cut boundaries cannot be assigned. Therefore, another fuzzy variable, D, which takes three linguistic values: 'hard', 'moderate' and 'easy', is assigned to represent the difficulty level of an MC test. The same fuzzy membership function given in Figure 6.2 may also be used for D.

Apparently, the variable P depends on SMS and D. However, once P becomes available, the suitable pedagogical action depends on all three variables. In other words, for a particular learner, after answering an MC test, the required level of PAS option could be determined by the values SMS, P and D. These causal relationships are represented by the directed acyclic graph in Figure 6.4. The fuzzy rules that govern these casual relationships are given in Table 6.2. It may be noted that the fuzzy rules closely reflect the decision making processes of a typical human mentor in different learning situations. For example, human mentors usually expect that the experts will perform well on easy assessments. Therefore, when an expert correctly answers an easy question, mentors will probably just confirm this answer and then move onto the next section (or offer some more difficult questions). In the meantime, if an expert fails in such a situation, the mentors will usually treat it as a slip-up, and give the learner another chance to correct their error. The first row in the given fuzzy matrix (Table 6.2) reflects the above situation. It is notable that the variable P is considered binary (it may be fuzzy or non-fuzzy) in Table 6.2. Another value

for P (medium) can be included in this model by just adding an additional column to the table.

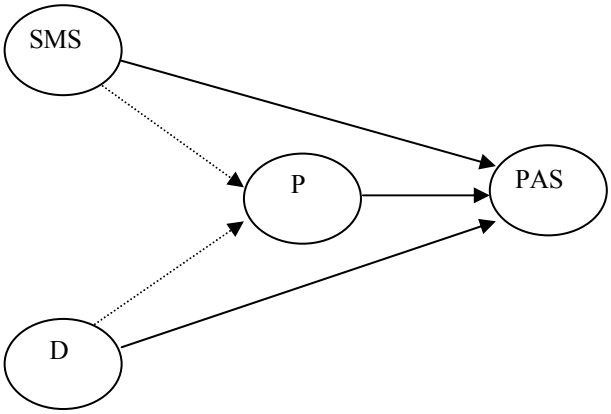


Figure 6.4 Causal Relations for PAS

Mental State (SMS)	Difficulty level (D)	PASw value P =Incorrect	PASc value P = Correct
Strong	Low	PASwL1	PAScL1
Strong	Moderate	PASwL2	PAScL2
Strong	High	PASwL3	PAScL3
Medium	Low	PASwL2	PAScL2
Medium	Moderate	PASwL3	PAScL3
Medium	High	PASwL4	PAScL4
Weak	Low	PASwL3	PAScL3
Weak	Moderate	PASwL4	PAScL4
Weak	High	PASwL5	PAScL5

Table 6.2 Fuzzy Rules for PAS

As previously stated, after a student has learned the material related to a sub-concept, LOZ assumes that they are in the corresponding basic mental state. In other words, this will be the first scaffolding stage related to the current sub-concept. To ensure the validity of this assumption, the system will give a relevant MC test. Depending on factors such as performance in the test and the difficulty level and the current strength of mental state, the system will select the appropriate level of PAS option (which provides appropriate feedback, and then selects the next suitable scaffolding stage). As previously assumed, SMS may be 50 (medium-100%) for the first basic mental state of a fresh main-concept. The effect of uncertainty is inevitable in the factors SMS and P. To lessen the effect of

uncertainty, in selecting appropriate pedagogical action, a well-defined fuzzy mechanism is incorporated in LOZ. The scaffolding process is now adaptable to individual learners.

In summary, since the critical response type tests are not used in the first phase, interactive problem solving support is not possible. Therefore, diagnosis at the micro-level is not necessary; however, macro-level diagnosis is performed at this time. The distracters for the MC tests are selected to match the potential misconceptions. Neither simple heuristics (as in the traditional CBL systems) nor a complex utility function (as in systems such as CAPIT (Mayo *et al.* 2001)) is used in this model. Instead, some natural fuzzy rules, which are almost parallel to the human tutor's decision making processes, are used. A well-defined fuzzy rule application process is designed to reduce the impact of uncertainty in the pedagogical action selection process. Nevertheless, whilst measuring performance, the time taken to answer a question is not taken into consideration. In reality, it can be expected that a good student will answer a question quickly (and correctly). The underlying fuzzy mechanism for PAS, used in this research, will be illustrated in the next section.

6.4.3 Fuzzification, Rule Propagation and Defuzzification

Initially, in the fuzzification process (Hopgood 2000), the crisp inputs for the variables SMS and D are converted to the corresponding fuzzy values, using the membership functions (Figure 6.2). If the fuzzy values are readily available (for example, from the previous calculations) the fuzzification process will not be necessary. Thereafter, in the rule propagation process, the relevant fuzzy rules are identified from Table 6.2, using the fuzzy values of SMS and D. These are then systematically applied to obtain fuzzy values for the output variable PAS. Finally, in the defuzzification process, the appropriate PAS level (or the crisp value for PAS) can be determined.

For example, a learner with SMS=65 failed a question with D=35.

In the fuzzification process, according to the fuzzy membership function given in Figure 6.2, the relevant degree of membership can be determined:

For Mental State (SMS):

$$\text{SMS} = 65 \rightarrow \mu_{Strong}(65) = 0.25,\ \mu_{Medium}(65) = 0.75;$$

and, for Difficulty (D):

$$\text{D} = 35 \rightarrow \mu_{Low}(35) = 0.25,\quad \mu_{Moderate}(65) = 0.75$$

The relevant fuzzy rules are given in the rows of Table 6.2. Since the answer is incorrect, it is only necessary to consider column P = Incorrect. For example, the first row represents the following rule:

IF Strength of Mental State (SMS) is Strong AND
Difficulty Level (D) of the test is Low AND
Answer is Incorrect
THEN
Learner needs a PASwL1 type pedagogical action – they might have made a careless mistake (slip), and therefore, they may be given a chance to try again.

Similarly, further rules can be formulated from the other rows of the Table 6.2

It can be noted, the fuzzy AND rule is: $\mu_{X\ AND\ Y}(c) = \min(\mu_X(c), \mu_Y(c))$.

and the fuzzy OR rule is: $\mu_{X\ OR\ Y}(c) = \max(\mu_X(c), \mu_Y(c))$.

For a complex if-then rule, using AND in the condition part, the confidence of the necessary part will be given by the minimum value of the confidence in the individual statements that make up the condition: similarly for OR rules, the maximum value will be given.

In the rule propagation, the following rules give the intermediate values for the degree of membership for PASw:

SMS(Strong) & D(Low) → PASw(PASwL1)

That is, $\mu_{PASwL1}(C) = min(0.25, 0.25) = 0.25,$

SMS(Strong) & D(Moderate) → PASw(PASwL2)

That is, $\mu_{PASwL2}(C) = min(0.25, 0.75) = 0.25,$

SMS(Medium) & D(Low) → PASw(PASwL2)

That is, $\mu_{PASwL2}(C) = min(0.75, 0.25) = 0.25,$

SMS(Medium) & D(Moderate) → PASw(PASwL3)

That is, $\mu_{PASwL3}(C) = min(0.75, 0.75) = 0.75,$

The resultant degrees of memberships for the variable PASw are;

$\mu_{PASwL1}(C) = 0.25$

$\mu_{PASwL2}(C) = max(0.25, 0.25) = 0.25$

$\mu_{PASwL3}(C) = 0.75$

Finally, in the defuzzification process, the crisp value C of PASw can be determined, using the fuzzy membership function given in Figure 6.3. However, at this juncture, the Learner model only needs to suggest the appropriate level of pedagogical actions for a particular learner and, therefore, the model only needs to select a PASw level between the five options. In other words, the Learner model needs to select a range, rather than a single numerical value for PAS. Therefore, in defuzzification, it is sufficient to select the PASw level (the linguistic value) with the highest degree of membership. If two PAS levels have an equal degree of membership, the highest PAS level will be selected.

This results in:

$\max(\mu_{PASwL1}(C), \mu_{PASwL2}(C), \mu_{PASwL3}(C)) = max(0.25, 0.25, 0.75) = 0.75$

The highest degree of membership is 0.75, and the corresponding fuzzy value is *PASwL3*

Therefore, the required PASw level is: *PASwL3*

If a learner with SMS=65 fails a question with D=35, the system proposes a third level pedagogical action for the incorrect answers (PASwL3). This means the feedback should include the explanation for each option, such as why it is correct or incorrect and later another MC test, at the same scaffolding stage, will be offered to the learner.

Alternatively, a complex defuzzification process may be used to determine the crisp value of PASw. Being the single numerical representation for the required pedagogical action for a learner in a particular MC test, this value may be stored in the Learner model for later usage. An illustration of the defuzzification process is included here for completeness. The Larsen's Product Operation Rule (Hopgood 2000) combined with *mirror rule at extremes* is used for the defuzzification process in this illustration. Initially in the defuzzification process, the fuzzy membership function (Figure 6.3) will be adjusted, based on the current possibilities (the calculated degree of membership). In Larsen's Rule, the membership functions are multiplied by their corresponding possibilities.

Figure 6.5 gives the adjusted membership functions, using Larsen's Rule, after applying the mirror rule at the extremes. At the boundaries (here at 0), the fuzzy set for PASwL1 is reflected on an imaginary mirror. At this time, the *centroid* method can be used to find the crisp value. In this example, the balance point along the fuzzy variable axis, for the composite shape of the three shaded triangles in Figure 6.5, will give the defuzzified (crisp) value.

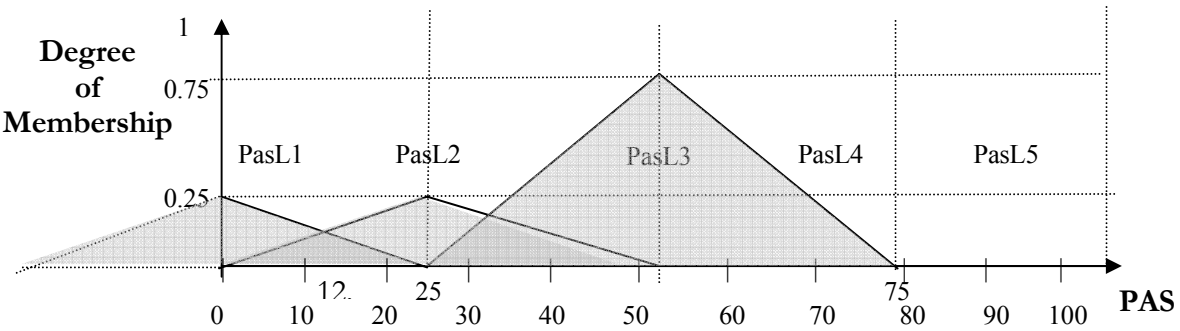


Figure 6.5 Defuzzification, using Larsen's Product Rule

In general, the centroid C along the X axis will be given by the following formula:

$$C = \frac{\sum x_i A_i}{\sum A_i}$$

Where, A_i is the area of a triangle and x_i is the distance of its centroid from the origin.

Since the triangles are equilateral, and the bases of the triangles are equal, the crisp value may be calculated by the following formula:

$$PASw = \frac{\sum x_i \mu_i}{\sum \mu_i}$$

Where, μ_i denotes the calculated possibilities (degrees of memberships)

Therefore, the crisp value for PASw will be:

$$\text{PASw} = \frac{0.75*50 + 0.25*25 + 0.25*0}{0.75 + 0.25 + 0.25} = 35$$

6.5 Predicting the Strength of Mental States

After a learning session, the strength of the mental state of a student, associated with the current topic, depends on how much they have learned during the session. In general, the degree of learning during a session varies with the learner's traits, abilities, motivation, conditions, past knowledge etc. In LOZ, a fuzzy variable L, that measures the degree of learning (in between two subsequent learner's interactions) from the system, will be kept explicitly. It may be assumed that the variable L also follows the same membership functions given in Figure 6.2 and takes three values: 'thoroughly', 'moderately' and 'scarcely'. The learner will usually interact with the system after learning a sub-concept or after receiving feedback, or in order to answer an MC test. Naturally, the variable L depends on the learner's general abilities and qualities. According to Vygotsky's (1978) zone of proximal development concept, the level of the general ability of a student to learn from social interactions (for the purpose of this research, it will come from the interactions with the system) plays an important role in the learning process. A variable GLA (General Learning Ability) is kept in the Learner model to denote the general learning ability of a learner. Similar to SMS, GLA can also take three values: strong, medium, and weak, and follow the same fuzzy membership functions given in Figure 6.2.

Based on the level of knowledge in the pre-requisites, the GLA for a learner may be initialised by some heuristic formulae. The following subjects are the pre-requisites for using LOZ for learning Object-Z: Discrete Mathematics, Object Oriented Concepts, UML model. The knowledge in Z, though not a pre-requisite, will definitely have a positive impact on the learning ability of a student. Initially, the knowledge level of the learner, for each subject mentioned above, may be measured by the system, or alternatively, the learner may be asked to provide their own estimations. The following heuristic formula is used to initialise GLA.

$$GLA_{initial} = \frac{7*Maths + 7*OO + 8*UML + 9*Z}{31}$$

The heuristic weights, as previously discussed, may be initially obtained by expert judgments, and can be modified later by machine learning techniques or by relevant empirical studies.

The degree of learning between interactions depends not only on the learner's general learning ability and traits but also on other factors, such as the length of time the learner actively spent on the learning process. If the variable T represents the variable length of time, it may take three values: 'much time', 'reasonable time', and 'little time' (see Figure 6.6). Basically, T is also fuzzy. The same linear membership pattern in Figure 6.2 may also be assumed for T. The fuzzy rules governing the causal relations expressed in Figure 6.6 are given in Table 6.3. As previously mentioned, measuring the time a learner spent actively on an on-line task is extremely difficult since learners tend to be frequently distracted from active learning. The research related to tracking eye-movements, in order to measure the active learning time, is still in its infant state (Anderson *et al.* 2001). To substantiate this difficulty, three constants are used in this research: 'minimum time' for experts, 'maximum time' for novices, and 'average time' for moderate learners. Alternatively, all three measures may be estimated for individual learners, using machine learning techniques (Stern *et al.* 1998). In extreme situations, when too much or too little time is spent on a lesson, the system may explicitly ask for the learner's assistance. In this situation, for example, the learner may provide an estimate for the time they have actively learned.

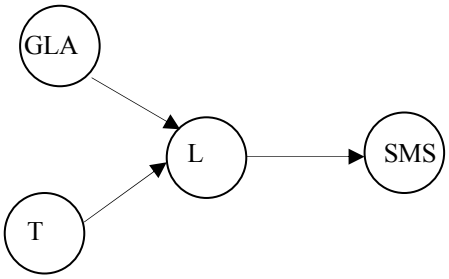


Figure 6.6 Causal Relations for SMS and L

Learn Ability (GLA)	Time Spent (T)	Degree of Learning (L)
Strong	Much	Thoroughly
Strong	Reasonable	Thoroughly
Strong	A little	Moderately
Medium	Much	Moderately
Medium	Reasonable	Moderately
Medium	A little	Scarcely
Weak	Much	Moderately
Weak	Reasonable	Scarcely
Weak	A little	Scarcely

Table 6.3 Fuzzy Rules for L (Degree of Learning)

6.5.1 Dynamic Relations

The variable SMS is dynamic, since its value changes over time and it is dependent on its previous values. Naturally, after learning a lesson, the measure SMS_{old} (related to the past knowledge level) will have a significant impact on the new knowledge level (SMS_{new}). Moreover, the learners may also forget or indeed learn something from other sources. Corbett *et al* (1992) use a type of Dynamic BN (Russell *et al.* 2003) approach but they do not explicitly mention it. Murray *et al* (2000) and Maya *et al* (2001) also use Dynamic BN for diagnosis and afterwards for decision making on PAS. Reye (2004) conducted extensive research on using Dynamic BN in CBL systems. In the context of Dynamic BN, the decision to use or not use a 'localised' model is, in computational terms, more significant than in BN. As Maya *et al* state,: "*An issue at this point was whether to use a model in which the constraints were independent of each other* [localized]*, as in Reye's model, or whether to allow (more realistically) any dependencies between constraints to be learned from the data"* (Mayo *et al.* 2001, p. 137). In a non-localised model, the entire knowledge network needs to be considered as dynamically changing over time. Since the Learner model in LOZ is a 'localised' model, the dynamic network will not be complex. Moreover, in this research a dynamic fuzzy mechanism, instead of Dynamic BN, is used in the main prediction mechanism.

Interestingly, all of the CBL systems discussed above update the model only at certain explicit events, such as the learner's interactions. Similarly, the variable SMS in LOZ will be predicted only with the learner's interactions, rather than at certain time intervals. However, there should be considerable learning activity (for example, learning a lesson or a detailed feedback) through use of the system between interactions. Moreover, the duration between interactions may also play a major role in a potential change in the learner's strength of mental state.

Reye discusses the difficulties of using time intervals for dynamic updating and states:

> *"It might be thought that we could better estimate the rate of remembering if we took into account the length of time between the $(n-1)^{th}$ and the n^{th} interactions, especially if such intervals may sometimes be large…..However, it is extremely difficult to create a psychologically-accurate model of human memory retention, especially when we don't know whether the student was studying the domain*

> *material independently during the same interval, i.e. while not using the ITS"* (Reye 2004, p.87).

Although a long duration away from the learning activity may cause memory lapse or there may be other learning from different sources, the time between interactions is not directly considered for dynamically predicting SMS. On the other hand, if the intervals are very small between interactions, it may be concluded that the memory decay and knowledge gained from independent study are negligible. Moreover, knowledge gained by using the system is also insignificant. Therefore, a time threshold is used in LOZ to avoid senseless updating, whenever the intervals between subsequent interactions are less. This means that, when the duration is negligible between adjacent interactions, the variable SMS will be considered static. However, it may be updated against any new evidence.

Figure 6.7 shows the dynamic causal relationship. The new-SMS will depend on both the old-SMS and the degree of learning between two subsequent interactions (L). The variable L depends on both the general learning ability of the student (GLA) and the time spent on learning (T). It is assumed that the variable GLA will not change during the learning of a main-concept. To simplify the situation in this study, Markov's first order chain, where the new value for SMS will depend only on its immediate preceding value, is assumed (Russell *et al.* 2003). It may be noted that impact of the difficulty level of the assessment on performance of a student is not shown in Figure 6.7 (see Figure 6.5). Table 6.4 shows the fuzzy rules for predicting new-SMS, using this dynamic relationship.

As previously stated, if two successive interactions are made within the time threshold, the variable SMS will not vary dynamically. In LOZ, this situation may arise when a learner answers two MC tests in succession, without spending too much time on the feedback. The nodes P2 and P3 in Figure 6.7 describe this situation. In this case, SMS will be considered static, and the old belief (SMS_2) will be simply updated twice, against the new evidence related to P2 and P3. In the context of Dynamic BN, Russell *et al* (2003) use the terms 'filtering', in their rain-and-umbrella example, for this updating process.

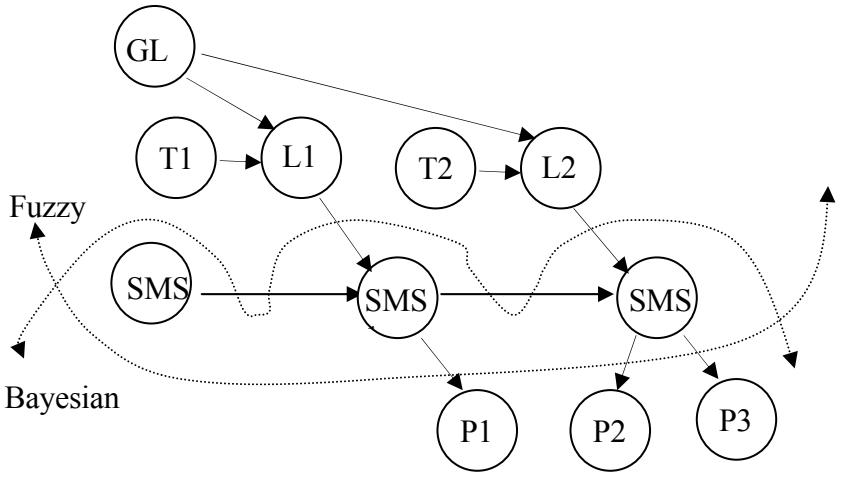


Figure 6.7 Dynamic Relations

SMS_i	Degree of Learning - L_i	SMS_{i+1}
Strong	Thoroughly	Strong
Strong	Moderately	Strong
Strong	Scarcely	Medium
Medium	Thoroughly	Strong
Medium	Moderately	Medium
Medium	Scarcely	Weak
Weak	Thoroughly	Medium
Weak	Moderately	Weak
Weak	Scarcely	Weak

Table 6.4 Fuzzy Rules for the Dynamic Variable SMS

Reye (2004) adopts the Dynamic BN approach (phase I and II) for Learner modelling in CBL systems. The approach used in this research is different from Reye's in three ways. Firstly and most importantly, the dynamic fuzzy model is used in this research instead of Dynamic BN. Secondly, if the duration between the subsequent interactions is less than a threshold, the dynamic variable (here it is SMS) is considered static and thirdly, if the duration is greater than the threshold, LOZ assumes that a considerable learning activity has been carried out using the system during the interval.

In summary, as a fuzzy model is used in LOZ to represent the dynamic causality related to a learner's strength in mental state, the underlying processes in the Learner model is more natural and resembles a human mentor's reasoning processes and understanding.

Moreover, a time threshold is used in the Learner model and, therefore, unnecessary (and erroneous) predictions are avoided.

6.6 Updating the Strength of Mental States

The performance of a learner answering a question may be considered as reliable evidence for the state of the corresponding mental state. The causal relationship given in the lower part of Figure 6.7 indicates that the performance of the learner depends on their strength of mental state (see Figure 6.4 also). This type of relationship is used in some ITSs to predict the performance before an assessment for task sequencing – to select a suitably challenging problem (Collins *et al.* 1996). This relationship is not used in this research for predicting performance but it is indeed used for the evaluation purpose as will be described in Chapter 9. Instead, the performance after assessment is used as evidence to update the variable SMS. It can be noted that, for the purpose of updating, BN theory is used. Fuzzy theory can be used in one direction only (predicting). Katz *et al* use fuzzy theory for their Learner modelling. However, they conclude:

> *"We might find that Bayesian inferencing techniques do a better job than ours at updating the student's knowledge state on variables within the revised network. One possible reason for this is that unlike our updating scheme, where reasoning takes place in only one direction Bayesian belief nets are quite capable of bi-directional reasoning"* (Katz *et al.* 1994, p.120).

In LOZ, as previously discussed, fuzzy theory is used in one direction for predicting both the suitable PAS after a formative assessment, and a new SMS value after a lesson (or a detailed feedback) is assimilated. In the other direction, BN theory will be used for updating the current SMS value, based on the performance of an assessment. Using probability to complement possibility is not an unusual process. As Berkin *et al* state, "*Possibility theory can also be considered as a form of probability theory with relaxed constraints. … Possibility also entails probability as one of its complementary elements*" (Berkin *et al.* 1997, p. 27). The subjective information, such as the competence level of a learner and the difficulty level of a question, can be efficiently analysed by using fuzzy theory, whereas BN is a powerful tool for updating prior beliefs, based on observable evidence. Therefore, combining both BN and fuzzy theories will give an elegant strategy

for uncertainty handling (Blair *et al.* 1999). In fact, in other disciplines, such as engineering, there have been many research attempts reported concerning combined strategies using BN and fuzzy theories to harness uncertainty (Taheri 2003).

Applying BN theory for real life problems involves two main challenges: tractability, and the selection of prior probabilities. Usually, real world problems involve complex belief networks. For an arbitrarily connected network, updating the belief in a BN may, in a worst case scenario, take time exponential to the number of nodes (called an NP-hard problem in computational mathematics (Cooper 1990)). However, there are efficient algorithms available for special structures, such as singly-connected nets. In this structure, any two nodes will be connected by only one path and, therefore, many pairs of nodes will be independent. Usually, in CBL systems, the potential BNs are not singly connected. However, the d-separation property (direction dependent criterion of connectivity) can be used to limit the belief-propagation (Pearl 1988). In LOZ, only a single level propagation is necessary for updating SMS, based on P and therefore, the related computational effort is negligible.

For updating purposes, since BN theory is to be used, the fuzzy model should be transformed accordingly. An approach that uses a straightforward and seamless transition process will be discussed first. SMS is considered as a concrete variable, which can be in any one of the three *distinct* states: 'strong', 'medium' and 'weak'. For a learner, the system may have different beliefs for each state. For a given SMS value, the fuzzification process can provide a different degree of membership for similar (but not the same) fuzzy states: strong, medium and weak. The degree of beliefs in each of the three distinct states can be initialised by the corresponding degree of membership of the fuzzy states. As a probability theory is involved, the above values may need to be adjusted to produce the total one.

For example, if Strength of Mental State (SMS) = 65 (see Figure 6.2);

$$SMS = 65 \rightarrow \mu_{Strong}(65) = 0.25,\ \mu_{Medium}(65) = 0.75; \quad \text{(Figure 6.2)}$$

Therefore, at this stage, the system's beliefs are:

P (SMS= Strong) = 0.25 and

P (SMS= Medium) = 0.75 and

P (SMS= Weak) = 0.0

Furthermore, it can be assumed that the difficulty levels of the questions are fixed (not fuzzy). A reasonable new set of ranges for the difficulty measure would be high (71-100), moderate (31-70) and low (0-30). Therefore, if a question has D=35, it is at the moderate level. The new causal relationship is shown in Figure 6.8 (right side). It can be noted that the new relations are just variations of the left side one, where three variables Pd, Pm and Pe are included to explicitly indicate the expected performance on different difficulty levels.

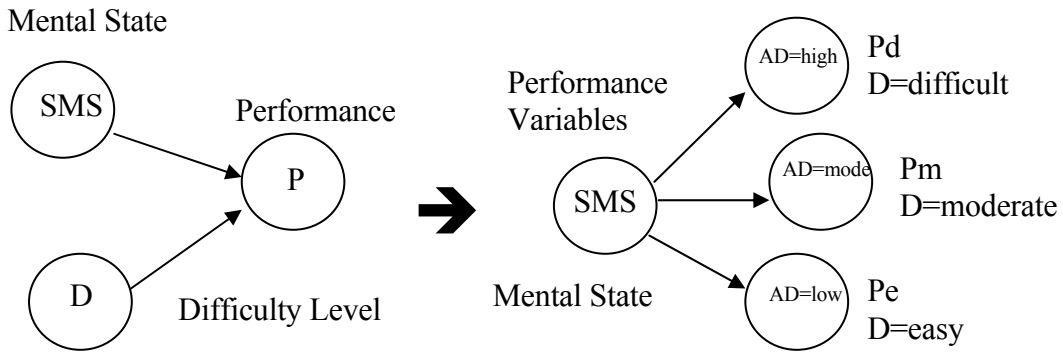


Figure 6.8 Modified Causal Relations

Table 6.5 shows the relevant conditional probability distributions for each difficulty level. SMS, with a certain level of strength is considered as the hypothesis H. SMS may take any strength value, strong, medium, or weak, with equal chances. The event that a MC test question, at a particular difficulty level, is answered correctly at the first attempt is denoted by E. As previously mentioned, one of the main drawbacks of using BN in uncertainty situations is the absence of accurate prior and conditional probabilities. Initially, the conditional probabilities may be suggested by domain experts.

The updating process is explained below using an example.

Presuming a learner with a Strength of Mental State (SMS=65) failed (P=incorrect) a question with difficulty D=35; and as before assume

D is a constant with (high (71-100), moderate (31-70) and low (0-30)).

Therefore, D= 35 ➔ D = moderate

Since the Strength of Mental State is 65 (SMS=65), by fuzzification:

P(strong) = 0.25,

P(medium) = 0.75

P(weak) = 0.0

The rows in Table 6.5 may be interpreted as rules. One of the relevant rules is (in the second row):

If the learner has a STRONG mental state
then he/she will answer a MODERATE level MC test
CORRECTLY
with a probability of 0.7

That is: P (E=mod-correct-first | H=strong) = 0.7

Or in other words: P (E=mod-incorrect-first | H=strong) = 1 – 0.7 = 0.3

Similarly, P (E=mod-incorrect-first | H=med) = 0.4

and P (E=mod-incorrect-first | H=weak) = 0.6

Strength (SMS) - H	**Difficulty (D)**	**P(E=correct-first |H)**	**Comments**
Strong	Low	0.9	1-slip
Strong	Moderate	0.7	Probability for Slip and Guess are included
Strong	High	0.6	
Medium	Low	0.7	
Medium	Moderate	0.6	
Medium	High	0.4	
Weak	Low	0.6	
Weak	Moderate	0.4	
Weak	High	(1/n) = 0.25	n - no of options (n>1)

Table 6.5 Conditional Probability Distribution (SMS&D)

The updated values will be calculated from the first principles

The general Bayes' rule is:

$$P(H \mid E) = \frac{P(H) * P(E \mid H)}{P(E)}$$

Now, let the evidence E be the FAILURE in a MODERATE level MCQ.

We have

$$P(H = strongh \mid E) = a * P(H = strong) * P(E \mid H = high)$$

$$P(H = med \mid E) = a * P(H = med) * P(E \mid H = med)$$

$$P(H = weak \mid E) = a * P(H = weak) * P(E \mid H = weak)$$

where $a = 1/P(E)$

$$\text{But, } P(H = strongh \mid E) + P(H = med \mid E) + P(H = weak \mid E) = 1$$

$$a \sum_{i \in \{strong,\ medium,\ weak\}} P(Hi) * P(E \mid Hi) = 1$$

$$a[0.25*0.3 + 0.75*0.4 + 0 * 0.6] = 1$$

$$a\,[0.075 + 0.3] = 1$$

$$0.375\ a = 1$$

$$a = 1 / 0.375$$

Therefore,

$$P(H = strong \mid E) = a * P(H = strong) * P(E \mid H = strong) = 0.075/0.375 = 0.2$$

$$P(H = med \mid E) = a * P(H = med) * P(E \mid H = med) = 0.30/0.375 = 0.8$$

$$P(H = weak \mid E) = a * P(H = weak) * P(E \mid H = weak) = 0.0$$

Therefore, in the light of new evidence that the learner failed in a moderate level test, the system's belief about the learner's ability is dragged down more towards moderate from high. Subsequently, if another updating (belief propagation) is necessary (if the learner interacted again within the time threshold), the above belief values may be used directly. Otherwise, as specified previously, the crisp value for SMS may be calculated, using Larsen's Production Rule. However, the defuzzification process is unnecessary here, since the above beliefs can be used directly as the corresponding degree of memberships in the subsequent prediction processes.

Although the above method for updating SMS is simple, it has a serious drawback. Since the updating formulae are based on multiplications, if the belief on a state initially assumes zero, it will be zero for all the subsequent updating.
For example, if SMS takes a value in the range 40 to 60:

P(H=Medium) =1 , and

P(H=Strong) = P(H=weak) = 0

In this case, the variable SMS will never be updated because all the other probabilities are zero. In general, if a state is certain, it will not be updated because the updating is done using a product operation. To overcome this problem, a new membership function may be introduced to avoid zero degree membership in fuzzification. However, this new membership function may be artificial and, moreover, keeping two functions for the same

variable (3-state SMS) may be confusing. Alternatively, instead of BN theory, certainty theory (Hopgood 2000) can be used for updating. In the certainty theory, the correction is done using addition operations, rather than multiplication. Instead of introducing another numerical method, to overcome this problem, the existing design may be slightly changed.

As previously stated, SMS is not fuzzy and will be updated using Bayes' theorem but it must be assumed that it can only be in any one of two states, strong or not strong, which means that the system believes, with certain probability, that the learner is reasonably strong in the corresponding mental state. In this interpretation, the crisp value of SMS is considered as the probability[6]. Let H denote the hypothesis that the learner is in a reasonable strong mental state (with certain probability). This hypothesis may be updated against the new evidence, as before, using Bayes' theorem. Alternatively, likelihood ratios may be used for updating beliefs (Russell *et al.* 2003). The expressions for likelihood ratios Affirms and Denies can be derived from Bayes' equations.

We have,

$$O(H) = \frac{P(H)}{P(\sim H)} = \frac{P(H)}{1 - P(H)}$$

$$P(H \mid E) = a * P(H) * P(E \mid H)$$

$$P(\sim H \mid E) = a * P(\sim H) * P(E \mid \sim H), \text{ where } a = \frac{1}{P(E)}$$

Therefore,

$$O(H / E) = \frac{P(H \mid E)}{P(\sim H \mid E)} = \frac{P(H)}{P(\sim H)} \times \frac{P(E \mid H)}{P(E \mid \sim H)}$$

$$= O(H) \times A \qquad \text{where } A = \frac{P(E \mid H)}{P(E \mid \sim H)}$$

Similarly, $O(H \mid \sim E) = O(H) \times D$ where $D = \frac{P(\sim E \mid H)}{P(\sim E \mid \sim H)}$

[6] Alternatively, if SMS is considered as fuzzy, a suitable membership function may be used to obtain the same effect. For example, for the crisp value SMS = 65, it is possible to select a suitable membership function that gives degrees of membership 65 for 'strong' and 35 for 'not strong'. After defuzzification, these values may be interpreted as probabilities. However, using two set of membership functions for the same variable may be confusing.

Let the event E be a MC test (of a particular level) which is answered correctly at the first attempt. The conditional probabilities for each difficulty level are given in Table 6.6. The variable *n* in the fourth column stands for the number of choices in MC tests. The likelihood ratios Affirms and Denies will be calculated using these probabilities.

H-SMS Reasonably Strong	Difficulty (D)	P (E|H)	P(E|~H) (n>3)	Affirms (A)	Denies (D)
T	high	0.6	1/n (0.25)	2.4	0.53
T	moderate	0.75	2/n (0.5)	1.5	0.5
T	low	0.9	3/n (0.75)	1.2	0.4

Table 6.6 Conditional Probabilities and Likelihood Ratios (SMS&D)

Consider the previous example:

Assuming that a learner with Strength of Mental State (SMS=65) failed (P=incorrect) a question with difficulty D=35.

Then, D= 35 ➔ D = moderate

The Strength of Mental State (SMS=65), as previously discussed, may be directly interpreted as a probability.

Therefore, SMS=65 ➔ P (H = Reasonably Strong) or P (H = T) = 0.65

Table 6.6 gives rules for updating. For example, the corresponding rule for the second row is:

If a learner has REASONABLY STRONG mental state,
then they will answer the MODERATE level MC test
CORRECTLY (AFFIRMS 1.5, DENIES 0.5);

The event E at this time is that the learner answered a Moderate level question correctly (at the first attempt). The event ~E will be that the learner failed at the first attempt.

We have, P(H=T) = 0.65

Therefore, O(H=T) = 0.65/0.35 = 1.86

According to the above rule, the fact that the learner FAILED in moderate level MCQ,

DENIES the likelihood by 0.5.

That is, O(H=T | ~E) = 1.86*0.5 (denies by 0.5)

$$= 0.93$$

New estimate for P(H=T) based on the new evidence ~E, will be P(H=T | ~E)

$$= O(H \mid \sim E) / (1 + O(H \mid \sim E))$$
$$= 0.93/1.93$$
$$= 0.48$$

Therefore, the belief that the learner is in a strong mental state is considerably dragged down from 65 to 48, in the light of new evidence that s/he failed in a moderate level test. It can be noted that now the SMS may take any value between zero and 100, depending on the likelihood ratio. However, since the fuzzy values and the crisp value of SMS are used for prediction and updating, respectively, there will be an overhead related to the fuzzification and defuzzification processes. Moreover, in general, the fuzzification process need not be an inverse of the defuzzification process and vice versa (refer to "*Hopgood's defuzzification paradox*" (Hopgood 2000)). Therefore, some valuable information may be lost during transition between fuzzification and defuzzification.

6.7 Initialising Strength of Mental States

Being locally-intelligent, the corresponding networks, for different main-concepts in LOZ, only have very limited links. As assumed in Section 6.3, for a new main-concept − after the relevant material is learned − the SMS of the basic mental state of the first sub-concept will be initialised to 50%. When moving to the next scaffold level, for the subsequent mental state in the series, the initial value for the SMS may be calculated as previously (Section 6.3). Alternatively, as the subsequent mental states are considered dependent (Figure 6.1), domain experts may provide initial estimates for relevant conditional probabilities. Therefore, BN theory may be used to obtain those initial values for SMSs. Similarly, after a sub-concept is learned, BN theory may be used to initialise the SMS of the basic mental state related to the next sub-concept.

SMS will be updated, possibly after each MC test. As previously stated for a sub-concept, after a mental state in the series is updated, if the SMS of the current mental state is greater than the SMS of the previous mental state, the SMS of all the previous mental states in the series that are less than the current SMS, will be updated by the current value.

The reason for this is that the SMS of the mental state in the lower rank should be greater than the SMS of any higher ranked mental states in the series. Put simply, if a learner can do something without any help, they should be able to do even more with extra help. However, if conditional probabilities are used for initialisation, the SMS values of the mental states in a series may be continuously updated backwardly. Moreover, for each visited mental state, the related SMS will be kept in the model. These values will not be updated, unless the same topic related to the mental state is visited again. In that case, SMS may be initialised to 50%, or to the previous SMS, depending on whichever is the larger.

In summary, possibility theory is used in the proposed Learner model for predicting both PAS and SMS. However, for updating SMS, probability theory is used (Figure 6.7). Except the SMS of the first mental state, associated with the main-concepts, all the other intermediate SMS values may be initialised using BN theory.

6.8 Determining General Learning Ability

The Learner model in this research keeps two running values, GLA and SMS, for a user. GLA may be interpreted as the system's belief about the student's general ability to learn any forthcoming issues. Similar to SMS, the variable GLA is also dynamic. A new value for SMS is frequently calculated. However, for GLA, a new value will be determined only after a main-concept is learned. This process, parallel to determining a new SMS, will be completed at two levels. Firstly, a new value will be predicted dynamically, and secondly, it will be updated in the light of new evidence. However, instead of fuzzy theory, Dynamic BN will also be used for the prediction part. Figure 6.9 describes the relevant causal relationships for the variable GLA, for updating purposes. Rather than linking GLA and SMS through L (see Figure 6.6), a direct causal relationship is assumed. At the conclusion of learning the current main-concept, the final SMS of the last sub-concept, will be used as evidence to update GLA. Initially, the new value GLA_{n+1} will be predicted dynamically from GLA_n. Thereafter, by considering SMS_{mn} as evidence, the current GLA_{n+1} will be updated (Russell *et al.* 2003; Reye 2004).

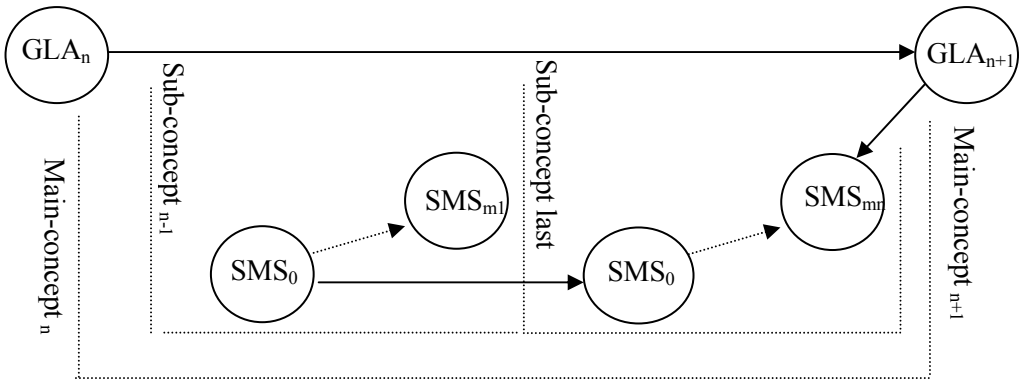


Figure 6.9 Dynamic Relations for GLA

Assuming both GLA and SMS are binary variables, they may be in any one of two states, 'sufficiently strong' or 'not sufficiently strong'. For both GLA and SMS, a score greater than 75 is considered as sufficiently strong. The conditional probabilities are given in Table 6.7. The attribute GLAn in the first column stands for 'Strong Learner' for n^{th} main-concept, and the conditional probability $P(GLA_{n+1} \mid GLA_n)$ in the second column is the probability of becoming a Strong Learner for $n+1^{th}$ main-concept after learning the n^{th} main-concept. The first row shows that, after assimilating a main-concept, a learner with sufficiently strong general learning ability will remain in the sufficiently strong state for the next main-concept with probability 0.9.

GLA_n	**$P(GLA_{n+1} \mid GLA_n)$**
T	0.9
F	0.2

Table 6.7 Conditional Probabilities (for Dynamic Relation)

For example, assume that the general learning ability of the user is 75 at the beginning of the current main-concept (that is, GLA_n=75), and also assume the strength of the last mental state of the last sub-concept is 70 after the final MC test (that is, SMS=70). The first measure GLA_n=75 will be directly used as the probability: that is, the system's belief is 75% that the learner has 'sufficiently strong' general learning ability (that means, $P((GLA_n=T) =0.75)$. However, the second measure SMS=70 will be used to determine the evidence: that is, in quantitative terms, to find whether the learner is in a sufficiently strong mental state or not. For example, since SMS=70, SMS is less than 75. Therefore

SMS=False. In other words, the learner does not have sufficient strength in the current mental state.

Firstly, in the dynamic prediction process (from GLA_n to GLA_{n+1}):

$$P(\text{GLA}_{n+1}) = P(\text{GLA}_{n+1} \mid \text{GLA}_n = T) * P(\text{GLA}_n = T) + P(\text{GLA}_{n+1} \mid \text{GLA}_n = F) * P(\text{GLA}_n = F)$$

$$= 0.9 * 0.75 + 0.2 * 0.25$$

$$= 0.73$$

Next, in the updating process (updating GLA_{n+1} based on SMS), Table 6.8 shows the relevant conditional probability and likelihood ratios. The calculations are similar to that of updating SMS.

H-Strong GLA	**E –Strong SMS**	**P (E| H)**	**P (E| ~H)**	**A**	**D**
T	T	0.8	0.3	2.7	0.29

Table 6.8 Conditional Probabilities and Likelihoods Ratios (GLA&SMS)

The corresponding Rule is;

IF a learner is in the sufficiently strong general learning ability,

THEN they get sufficiently strong strength in the final mental state

(AFFIRMS 2.7, DENIES 0.29) (Figure 6.8 & Table 6.8)

The current value for GLA is 73 (after dynamic prediction):

That is, $P(GLA_{n+1}=T) = 0.73$

Let the hypothesis H be that the learner has sufficiently strong general learning ability with the probability 0.73.

That is, $P(H) = 0.73$; but we have $O(H) = \frac{P(H)}{P(\sim H)} = \frac{0.73}{0.27} = 2.7$

As before, since SMS=70 (<75), learner does not have sufficiently strong strength in the final mental state. Therefore SMS=False.

Let the event E be SMS= False: this can be used as an evidence to update the current GLA.

The fact that the learner is not sufficiently strong in the last mental state denies the likelihood that the learner is sufficiently strong in the general learning ability necessary for future lessons, by 0.29.

$$O(H \mid\sim E) = O(H) \times D$$
$$= 2.7*0.29$$
$$= 0.78$$

Therefore, the new estimate for $P(GLA_{n+1})$ will be;

$$P(H \mid \sim E) = O(H \mid \sim E) / (1+ O(H \mid \sim E))$$
$$P(GLA_{n+1}) = 0.78/1.78$$
$$= 0.44$$

The system's earlier belief that the user is a strong learner is considerably dragged down from 73% to a lower value 44% in the light of the evidence that s/he failed to get sufficient strength in the last mental state.

Maintaining optimal Learner models with uncertainty handling strategies is challenging. In the next section, the process of externalising the Learner model will be discussed. It is expected that this action, together with many other advantages, may significantly improve the efficiency of the Learner model.

6.9 Opening Learner Models

In Chapter 3, various approaches and their related advantages and the limitations of externalising Learner models were discussed. Initially, the idea of opening Learner models to the learners was suggested by Self (1988), in order to share the burden in designing and maintaining adaptive Learner models. However, the current trend in research towards opening Learner models reflects contemporary learning theories. The constructivist and situated learning theories suggest that instructional design should include strategies for enhancing collaborative learning and meta-cognitive activities such as reflection. Furthermore, opening up a limited part of Learner models to peers and mentors has also been given much attention in recent research. This helps learners not

only to estimate their own abilities but also to compare these abilities with their peers, In addition, by providing facilities for expressing their views and concerns to their peers and mentors, the act of opening Learner model strengthens the link between mentors and learners in automated learning systems.

If a learner feels that the system is not providing sufficient feedback (it is probable that the system ranked the learner higher than they should be), one reason for this might be that the learner initially over-estimated their ability in the pre-requisite lessons. For example, if a learner with a high SMS score (for example, 90%) failed in a medium difficulty level question (for example, 55%), the system may assume it is a slip-up, and will just inform the user why the selected answer was incorrect. Finally, it will give another opportunity to correct the error. Actually, in the above situation, the learner requires detailed feedback. To understand the reason for this problem, the system provides facilities to inspect the state of the Learner model and the decision making process. The explanation will be given to the learner, using every-day terms (for example, "Very Strong" instead of "90% Strong"). Later, the learner, being equipped with the information that the system has ranked them incorrectly, may wish to alter the Learner model. The usage of a fuzzy model makes it easy to visualise the system's beliefs about a learner's different strengths. The visual models may be annotated, using the related linguistic terms used in the fuzzy model. Explaining a fuzzy model in natural language is easy, since the model uses the same terminology.

For a particular MC test, assuming the difficulty level (D) is a constant and, after an answer is given, the performance (P) is also a constant (Figure 6.4). Therefore, altering the PAS level will only affect SMS. The PAS level decreases from five to one, as SMS increases from weak to strong. An experienced learner may easily verify that, at a given scaffolding stage, if their answer is incorrect, they used to receive detailed feedback when their knowledge level was weak. Therefore, they will know what to change and then what to anticipate and they can easily understand the possible impact of their intended changes on the subsequent behaviour of the system. For beginners, however, judging their required level of feedback PAS (effect-variable) would be easier than estimating their own strength of mental state SMS (cause-variable). Therefore, the interface (screen shots are shown in Chapter 8) allows the learners to use any one of both options to alter the SMS directly or, for an inexperienced learner, to set the desired PAS level so that the

SMS rate will be automatically adjusted. To give visual cues, the numerical values will be automatically annotated by relevant bar charts.

Being a fuzzy model, it is straightforward to visualise the impact on the system's belief, due to any adjustments made by a learner. If learners feel later that they had rated themselves incorrectly, they may simply try fresh values. This trial-and-error feature not only reduces the burden of the Learner model, but it also allows the learner to have some control over the system's decisions. These facilities, in turn, encourage the learners to reflect on their own learning progress.

Moreover, learners, who want to take the ultimate responsibility for their learning process, may wish to investigate the underlying fuzzy mechanism. As the Learner model is based on fuzzy logic, the rule application process can be seamlessly described in natural language. For a learner, it will be relatively easy to understand the fuzzy mechanism, compared to other numerical approaches, such as BN. The enthusiastic learners, if encouraged to use this facility, would be able to significantly enhance their self-confidence.

This system also keeps relevant information about the current and past learning activities of a learner. Learners can view their past performance, related to the concepts they have learned to date. They may check their stored SMS values, related to past learned concepts and, if they so wish, they may change it. However, to improve the system's belief, the learner may need to convince the system. Bull (2004) terms this 'negotiating', which means that the system may offer some tests to the learner, in order to give them an opportunity to prove their ability. Negotiating is also possible if (genuine) learners want to degrade themselves. The learner may revisit a recently attempted MC test, and attempt it again, or process the feedback in different detail. They can also go through the past scaffolding processes and critically analyse their own decisions in MC tests.

Finally, if a group of students are involved, the learner may be able to compare her/his performance against the best, worst and average cases. This feature is also used by some other open systems, for developing reflection habits in students (Kay 2000). For the purpose of academic learning, going beyond situated cognition, Laurillard (2002) argues that multiple contexts are not sufficient and that learners need to be engaged not only with

their own experience, but also with knowledge derived from someone else's experience. By opening the Learner model to peers and mentors, a facility is provided to the learners to decontexualise their knowledge, not only in multiple contexts but also through social experience.

Other than opening the Learner model, the mentoring processes can also be revealed to the learners. Mentor related issues, such as the scaffolding stages, the number of scaffolding steps, marking schemes and feedback strategies can be opened. The learner may make use of this opportunity to harness the learning environment to suit their traits and abilities. The next section will discuss this issue in detail.

6.9.1 Opening Mentor Model

In LOZ, the Learner model has no explicit control on deciding the mentoring actions. The mentoring actions are hardwired in the tables that describe the different levels of PAS tasks. After a test, the system may decide whether to allow the learner to stay at the same level, move onto the next higher level, or to move to the next lower level. This decision is made based on different factors such as, performance of the student, difficulty level of the question and rank of the learner. In an extreme case, the system may allow the learner to move two scaffolding levels upwards. However, for this to happen, the system should rank the learner too high.

The scaffolding levels are selected to suit potential lower level learners. This arrangement may cause boredom for some learners (particularly, for gifted students). Different learners have different learning abilities and, therefore, it would be beneficial to allow each learner to decide their own scaffold steps that best meet their needs. Moreover, a gifted learner may skip the initial steps and also they do not need to start from the basic scaffolding state.

In LOZ, before or after learning a lesson, the learner can inspect the levels and types of scaffolding process, and s/he can also modify this process for a lesson (initially some learners may perceive the scaffolding levels simply as different difficulty levels). For example, a lesson 'Visibility List' in LOZ has three scaffolding stages (Figure 6.1). A gifted learner may change the starting stage to three and, therefore s/he can avoid the easy

early stages. Some lessons may have more than five stages. The incremental steps can also be altered. This facility will help learners to control their learning process, and in turn it will help them to develop their meta-cognitive abilities. If learners feel later that they had selected inappropriate start levels or stage steps, they may revisit the lesson and assign new values for themselves.

The learner may also be allowed to decide their type of feedback and its timing. In the context of confidence based MC tests, the marking scheme and intermediate calculations can also be revealed to the learners. In extreme cases, if they can make use of it, the learners may be given the entire design and coding material.

6.10 Summary and Conclusion

A fully user controlled environment is beneficial for diligent students who could take full responsibility for their learning process. In general, novice students need specific support and, therefore, Learner models are inevitable. Learner models may be designed with different complexity levels. In many cases, locally-intelligent Learner models are sufficient. In this research, a numeric measure (SMS) is kept, to gauge the competency level of a learner on a specific topic. Meantime, a sequence of pedagogical options by combining and ranking feedback and curriculum sequencing tasks (PAS levels) is designed to match with usual teachers' actions. A simple Learner model can be designed, using a one-one mapping from SMS to PAS levels. However, this model is not efficient. For instance, an expert learner may make a slip-up, or a novice may guess correctly. Since the evidence is limited, assigning credits between various probable causes will be difficult.

A mathematical approach based on both possibility and probability theories is used in this study. The aim of this approach is to reduce the impact of uncertainty. A set of fuzzy rules is designed for pedagogical decision making. These rules reflect the typical human tutor's decision making processes. Being locally-intelligent, the causal net is simple and easy to update. Unlike BN theory, Fuzzy logic cannot be used for backward prediction. Therefore, BN theory is used to update the net.

A knowledge is dynamic in nature – it may independently change with time. Fuzzy rules are also designed to cover this dynamic aspect. A time threshold is used to avoid senseless updates. In addition to the variable SMS, another numeric measure (GLA) is used in this research to estimate the general traits of a student (independent to the current topic learned). GLA is also naturally changes with time. Dynamic BN is used to predict and update GLA also.

The Learner model may be opened to students, peers, and mentors. It has many advantages: for example, discrepancies in the Learner model may be identified and corrected at the early stages. Insufficient bandwidth for evidence causes uncertainty. Learners can provide their own estimate. Enhancing meta-cognitive abilities of the students is another key advantage. Opening a fuzzy Learner model is comparatively easy and seamless. Moreover, opening the scaffolding process, particularly allowing the learner to inspect and modify the scaffolding stages and steps, also has pedagogical advantages. A prototype for LOZ has been developed to demonstrate the proposed features described in Chapters 5 and 6. This prototype will also demonstrate some facilities related to externalising the Learner model. The key issues related to creating a prototype will be discussed in the next chapter.

Chapter 7

Prototype – Design and Implementation

7.1 Introduction

This chapter discusses the design and development process for the prototype created in this research. The purpose of the prototype is to evaluate the concepts proposed mainly in Chapters 5 and 6. The prototype is designed using an OO approach in an iterative fashion under JBuilder visual environment. The executable prototype is later evaluated using postgraduate students. Chapter 8 discusses the evaluation process and results in detail. Since it is an evolutionary prototype and developed using an OO approach, the design and code can be easily modified or extended.

Section 7.2 provides the scope of the prototype. Section 7.3 discusses some key implementation decisions. Section 7.4 describes the high level architecture of the system. The domain model design is outlined in Section 7.5. The primary focus of the prototype is the Learner model. Section 7.6 describes the design and implementation of the Learner model. The fuzzy model is discussed in Section 7.7. Though both the domain and Learner models contain most of the persistent data structures in the system, only the Learner model will be updated more frequently. Section 7.8 discusses the Mentor model design, which is mostly responsible for the dynamic functionality of the system. Database design is given in Section 7.9. The interface design of the prototype is outlined in Section 7.10. Design relevant to opening Learner model is discussed in Section 7.11. As mentioned before, the Refinement unit in phase-I includes only a one-one mapping from specifications to respective Java code, and a simple matching process. It will not be discussed further in this chapter.

7.2 Scope of the Prototype

Software prototypes may be developed for different purposes, such as elicitating requirements, evaluating interface design and for inspecting certain functional features (Sommerville 2001). The prototype in this research is developed to serve two purposes; firstly, to determine whether the proposed ideas in this research are technically feasible and secondly, to verify that the implemented artefacts are practically useful. Due to the time constraint, the interface design of the prototype is not a major focus.

The following guidelines determined the required characteristics (or the boundary) of the prototype.

- It should cover at least one main-concept
- The FOPSI model should be implemented
- The fuzzy based Learner model should be implemented thoroughly
- The interface must not hinder learning activities.
- The MC test format can be used for formative assessment (essay type questions that require free-form answers may not be supported)

The first two guidelines ensure that the system covers sufficient portion of the learning material for this purpose. The fuzzy based locally intelligent Learner model discussed in Chapter 6 is the key proposal of this research; hence the third guideline. Obviously, for an interactive tutoring system, effectiveness of the interface plays an important role in user satisfaction and acceptance. The fourth guideline ensures that the interface issues will not adversely affect the learning process, and therefore, the effect of extraneous variables in the evaluation process could be minimized.

An assessment process using essay type questions would be preferable to using MC test questions; however, due to time constraints the assessment included in the prototype is limited to MC tests. Object-Z specifications include complex mathematical notations as well as graphical representations. Therefore, providing on-line facilities for free form answering for a subject like Object-Z would be difficult and time consuming. Hence, it is not feasible for this project. However, a full-fledged learning system should support essay-type assessment schemes; it should include at least a graphical editor capable of

creating Object-Z specifications, a syntax parser to check and parse the input, and also a semantic checker to provide context sensitive help during editing sessions.

7.3 Implementation Issues

The Java programming language was selected for implementation. It is a secure architecture-neutral object-oriented language that can be distributed and portable (Gosling *et al.* 1996). JBuilder, a visual development environment for Java, is used in this project. The Swing components in JBuilder (a set of Graphical User Interface (GUI) components) are built on top of Abstract Window Toolkits (AWT). However, Swing offers many advantages over AWT. First, Swing components are more portable and can be made to look like Windows, or any other look and feel for which appropriate classes are present. Second, they follow the MVC pattern (Model View Controller). In addition to basic Swing components, JBuilder also includes a value-added component called dbSwing for database access.

JDatastore, an object–relational database, is used in this project. It is a SQL database with a standard JDBC local or remote driver. It was written entirely in Java and fully integrated with JBuilder. JDatastore can also be viewed as a portable file system, because, a single physical file in JDatastore can be used to hold different items such as the data, files and any other object (JDatastore 2006). However, creating and editing tables in JDatastore is a tedious task. Being a new system, the front-end of JDatastore (called JDatastore Explorer) has not got a smart offline interface for manipulating tables (see Figures 7.1& 7.2).

7.4 System Architecture

The overall architecture of the proposed system was described in Section 5.5. The prototype is developed based on a layered architecture. Figure 7.3 describes the horizontal and vertical sub divisions of the system. The arrows indicate the possible navigational direction of the objects in different layers. The objects related to one layer can only communicate with the objects of the neighbouring layers. In other words, each layer depends only on the layers adjacent to it. Therefore, for example, the application layer may be reused with different databases and alternate interfaces. However, performance of the system is severely impaired as higher level modules need to penetrate through several

layers to access the data base. Being an interactive event-driven teaching system this drawback needs to be addressed in later stages before finalising the interface of the system.

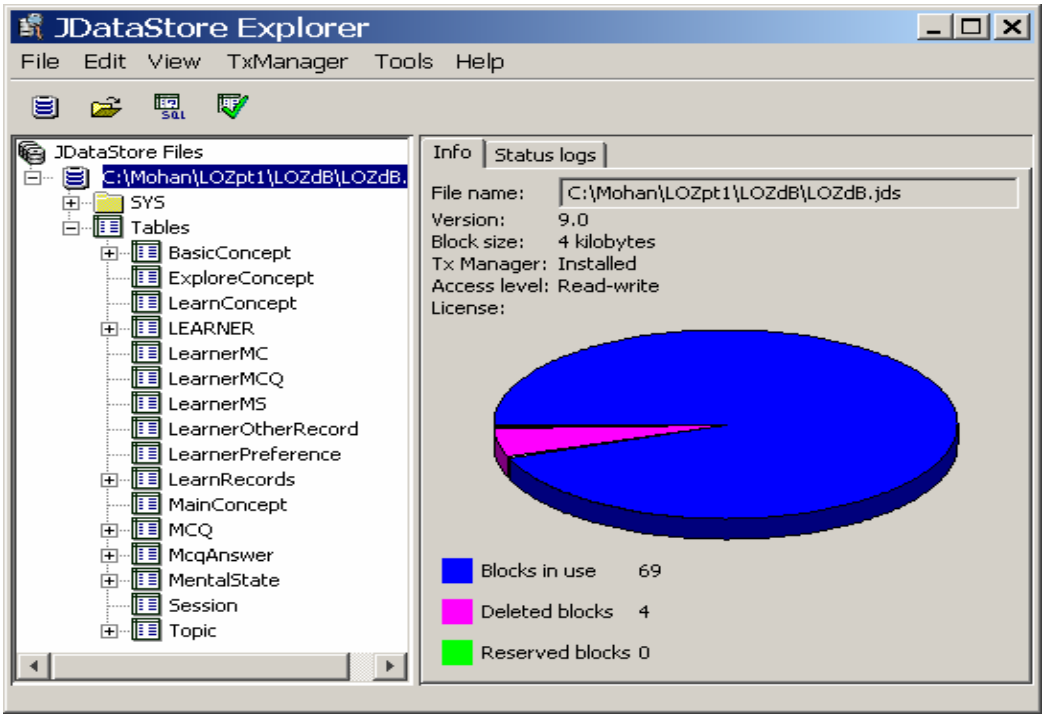


Figure 7.1 JDatastore Explorer Environment

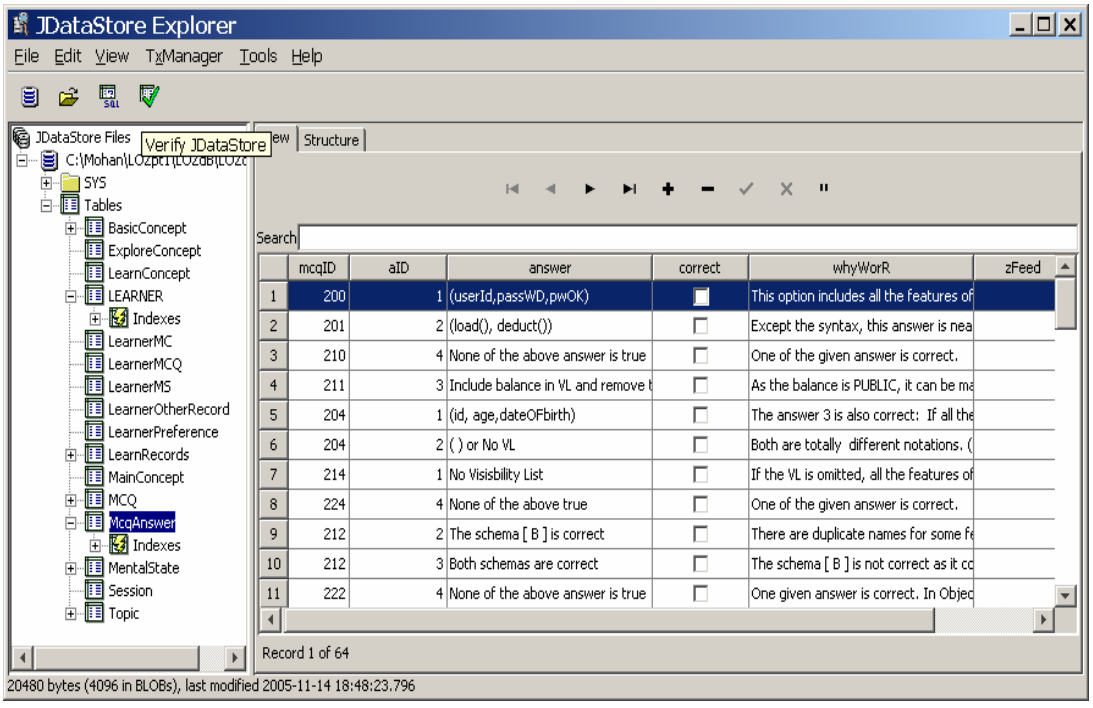


Figure 7.2 Interface for Editing Table in JDataStore

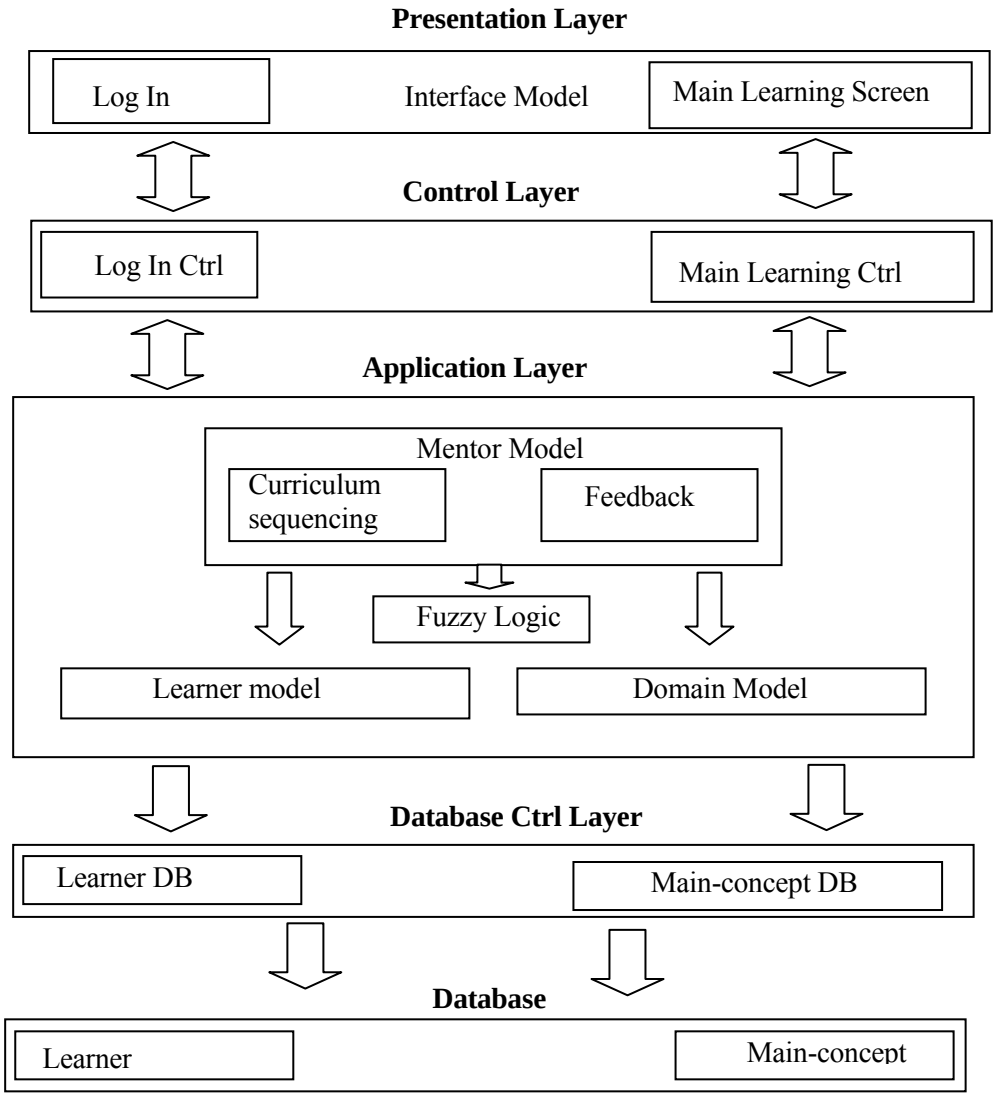


Figure 7.3 Layered Architecture

Figure 7.4 shows a sequence diagram of the log-in process for a returning student on a happy-day scenario (if everything goes right). This diagram clearly shows the message passing mechanism between different layers.

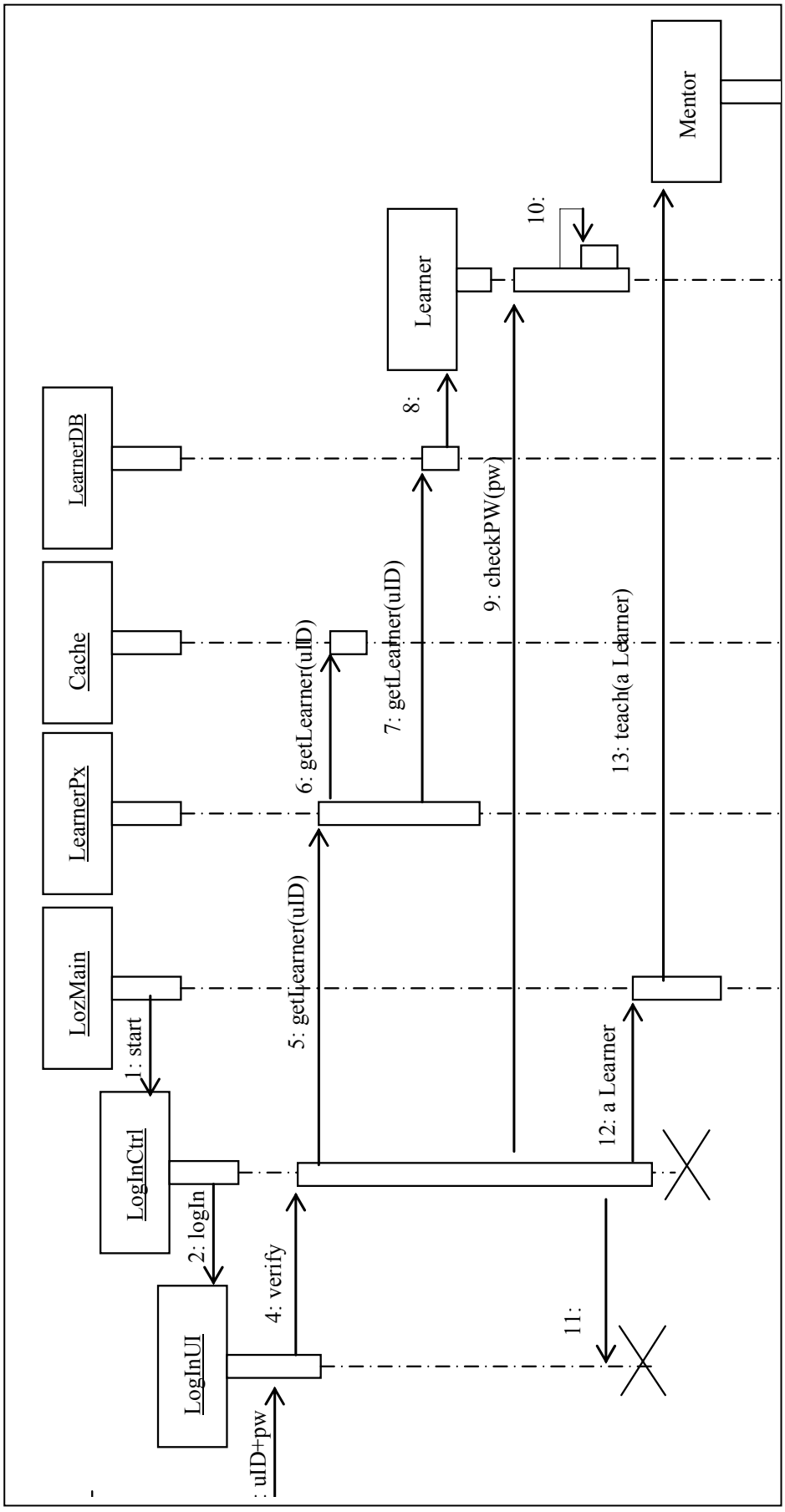


Figure 7.4 Log-in Process: A Happy Day

7.5 Domain Model Design

The course is divided into a collection of lessons. Each lesson has several topics. Each topic has a series of main-concepts. Each main-concept is associated with a pre-requisite concept, a sequence of sub-concepts, and a set of exploration concepts (see Figure 6.1). The foundation of the domain model is a basic-concept. Other relevant structures are inherited from the basic concept. Figure 7.5 gives a UML class diagram which specifies the partial structure of the domain model related to the concepts and their constituents. For example, a main-concept in the domain model is "Visibility list specifies the accessibility of the features of a class". A set of lesson material is prepared to cover this main-concept (see Appendix B). It includes materials relevant to sub-concepts, pre-concept, exploration concepts, MC test questions, distracters, mental states, and feedbacks.

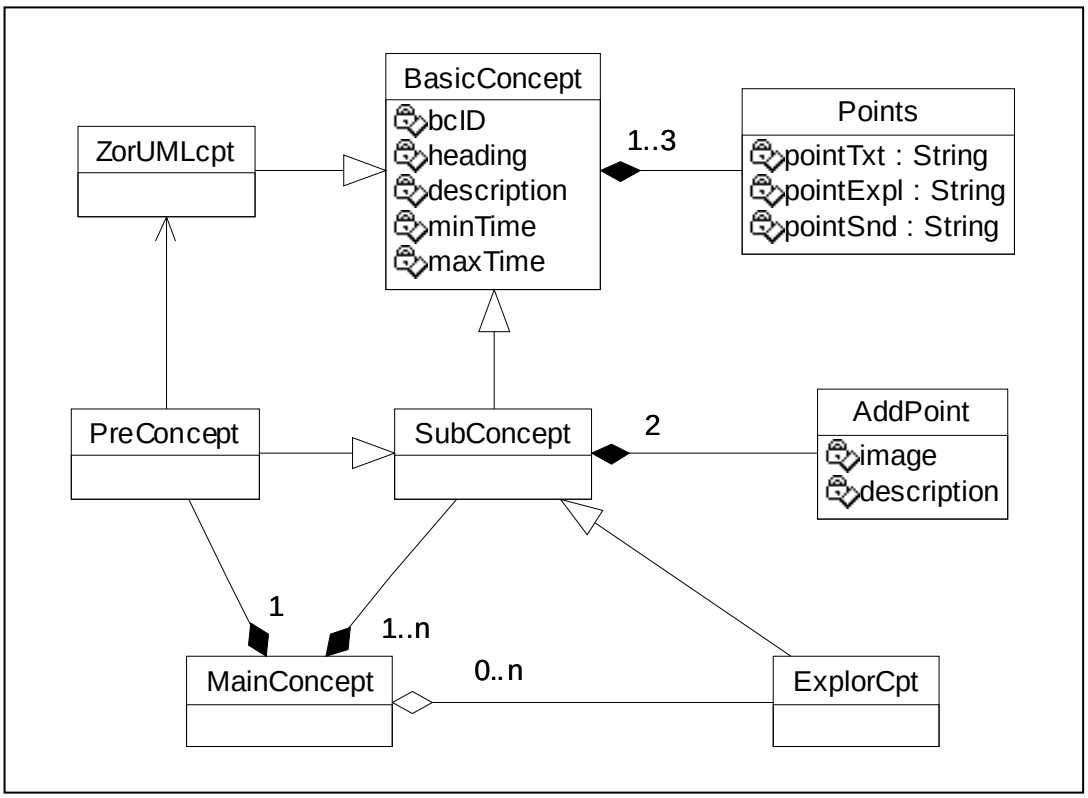


Figure 7.5 Class Diagram for Domain Model - Concepts

A basic-concept contains a heading, its description, and three points. It also includes two estimates for minimum and maximum time requirement to learn this concept. Each point contains a textual item and the corresponding explanation in both text and voice. A sub-

concept is essentially a basic-concept, but additionally it includes two structures meant for a notation and an example. Both notation and example structures consist of an image and a textual explanation. For an example, a screenshot in Figure 7.6 shows the attributes of a sub-concept as displayed on the screen. There are some relevant materials included as basic concepts to cover UML and Z fundamentals.

The UML class diagram in Figure 7.7 specifies a portion of domain model related to mental states. Each sub-concept is associated with a series of mental states and a collection of misconceptions. The series contain basic, final, and intermediate mental states (see Figure 6.1). The basic and intermediate states are included to support scaffolding process in the first phase. Each mental state is a scaffolding stage. There will be several MC tests for each scaffolding stage. An MC test has four answer options. The distracters are associated with one or more misconceptions, and different levels of feedback concepts. A screen shot in Figure 7.8 shows an MC question and answering interface.

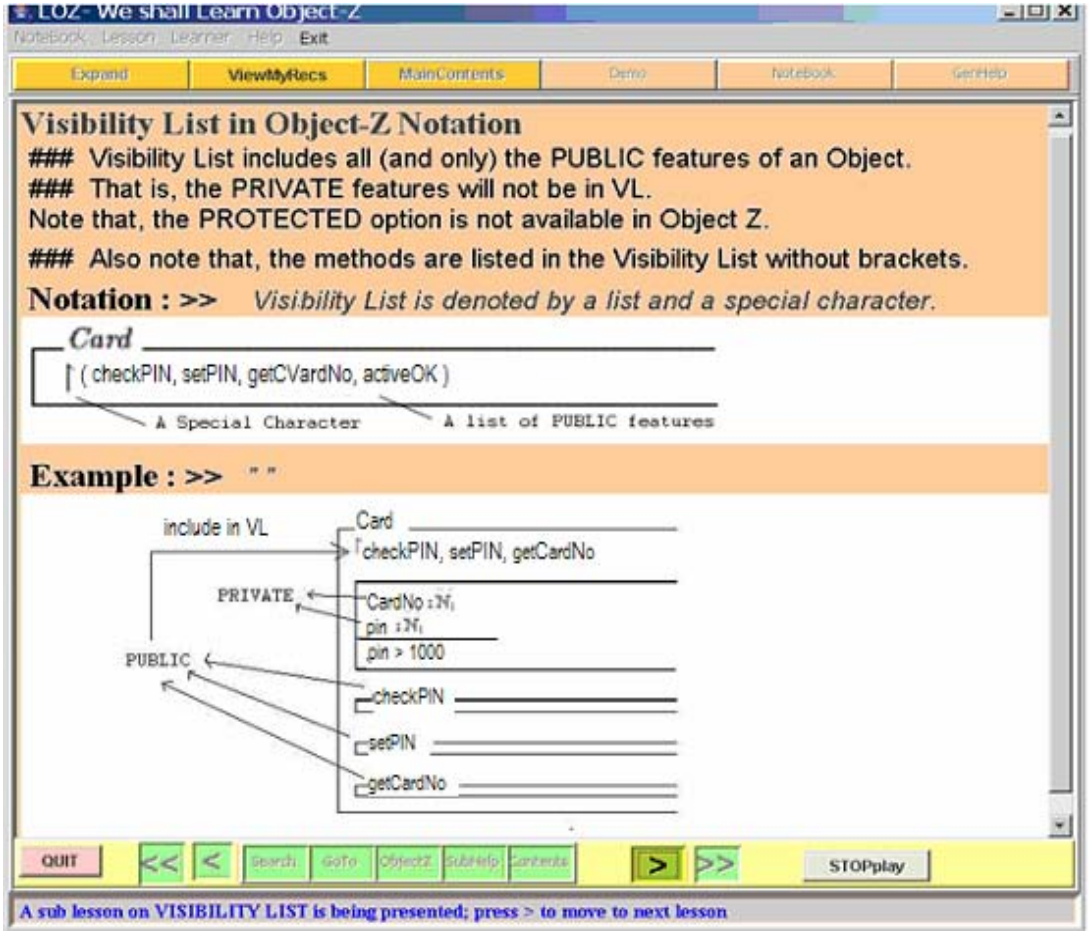


Figure 7.6 A Sub-concept: Visibility List

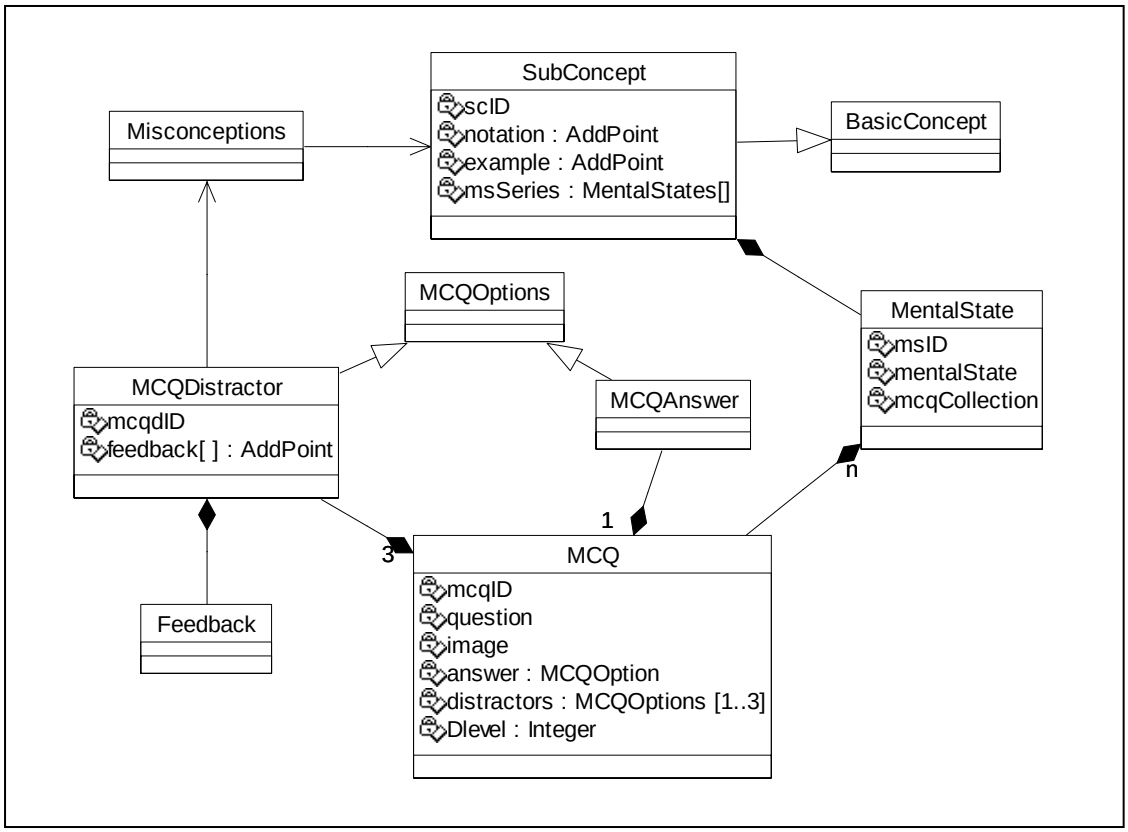


Figure 7.7 Class Diagram for Domain Model – Mental States and Concepts

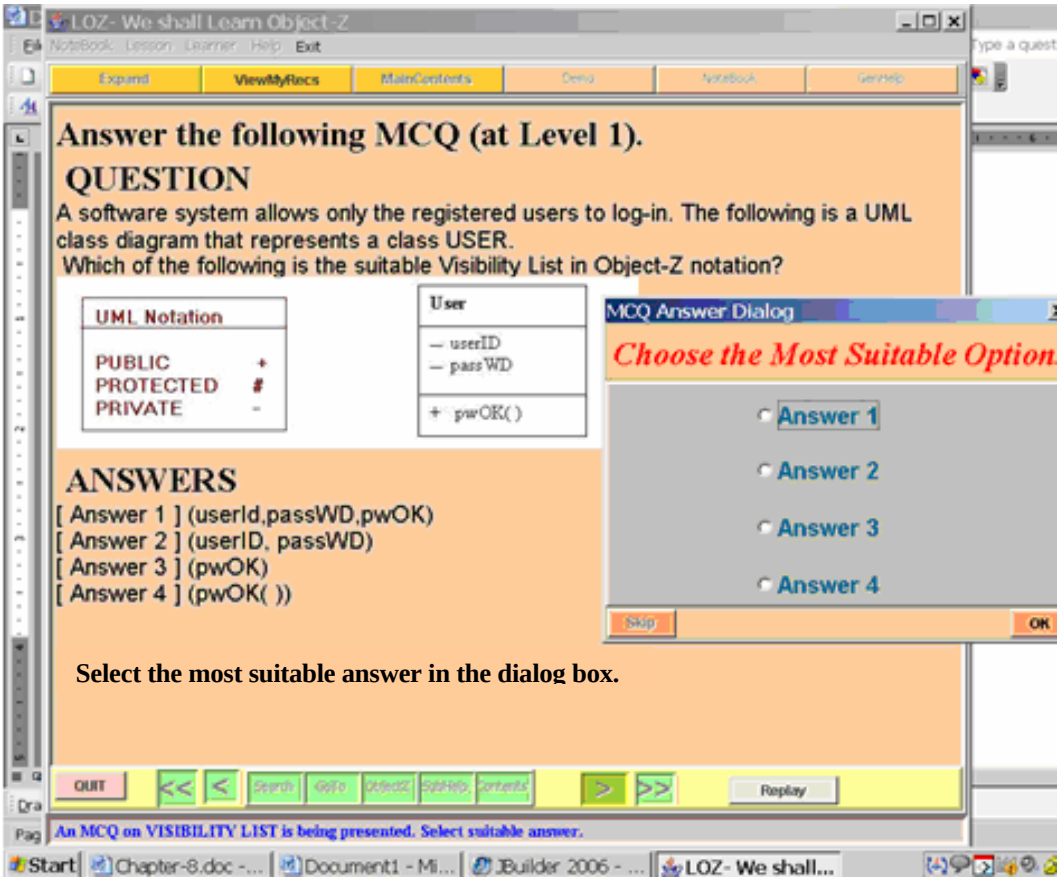


Figure 7.8 A MC Test & Answering Interface

7.6 Learner Model Design

For each learner, the system keeps relevant information in order to provide adaptive help when required. A new learner needs to provide a user name, password, and their basic preferences. They also need to estimate (or they may allow the system to perform the estimation) of their knowledge level of the pre-requisite subjects such as Mathematics and UML notation. Though not a prerequisite, the knowledge level of Z notation is also required. Based on these values, an initial estimate of their general learning ability (GLA) will be determined (Figure 7.9). A main-concept may be learned in more than one session. A session is a length of time a learner uses the system. A time threshold is used to discard small durations. The GLA will be considered constant for a particular main-concept. It will be updated at the end of learning a main-concept using Bayesian calculations considering the latest SMS value as new evidence.

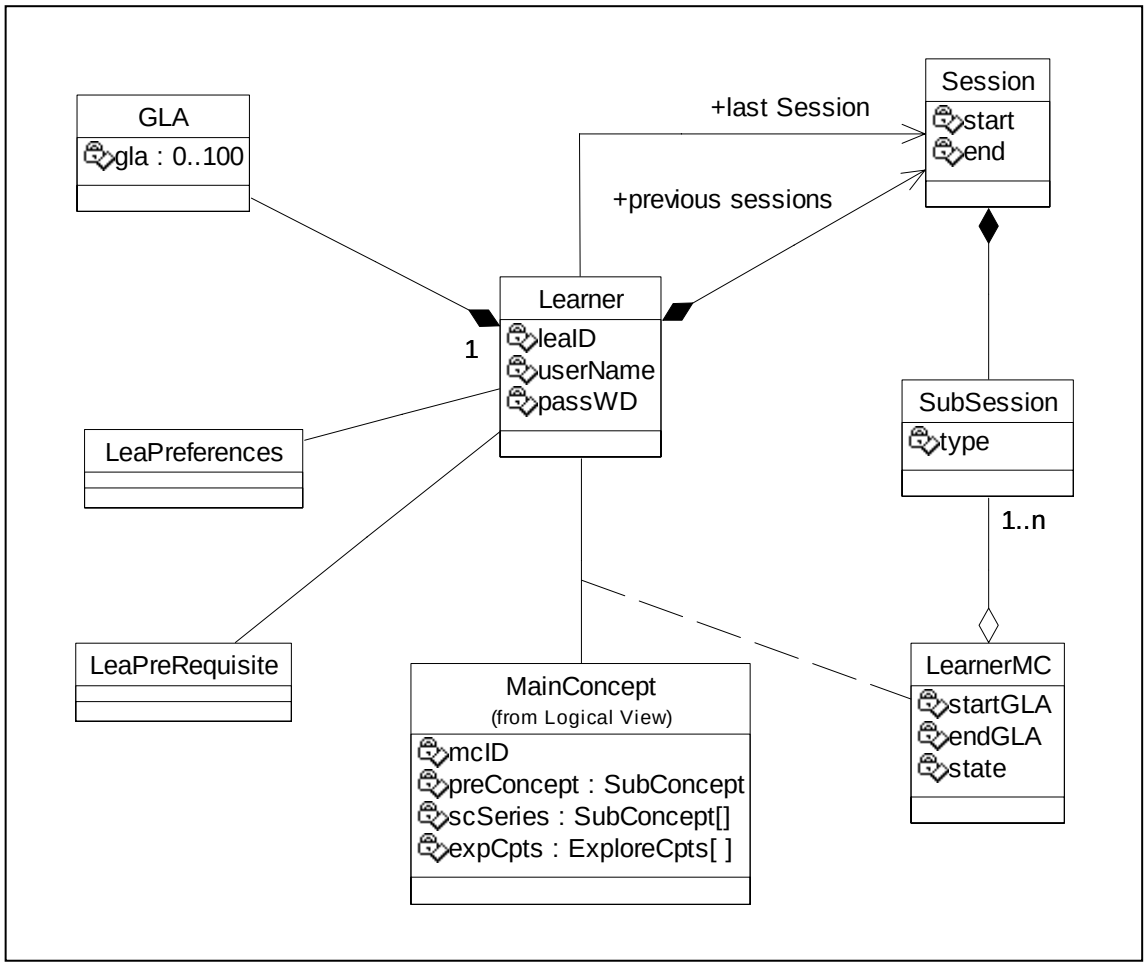


Figure 7.9 Class Diagram for Learner Model – Learner and GLA

The system keeps information about the student's interactions as a series of learning sessions. The relevant details of all the key interactions made in between their log-in and log-out are recorded. A session may contain many sub-sessions. Different learning activities such as, learning the precondition materials, learning a sub-concept, doing an MC test are considered as different sub sessions. The UML class diagram in Figure 7.10 specifies a partial structure of the Learner model. For each learner and for each mental state the corresponding SMS value will be kept in the Learner model. This value will be updated based on the performance of a learner on different levels of MC test items.

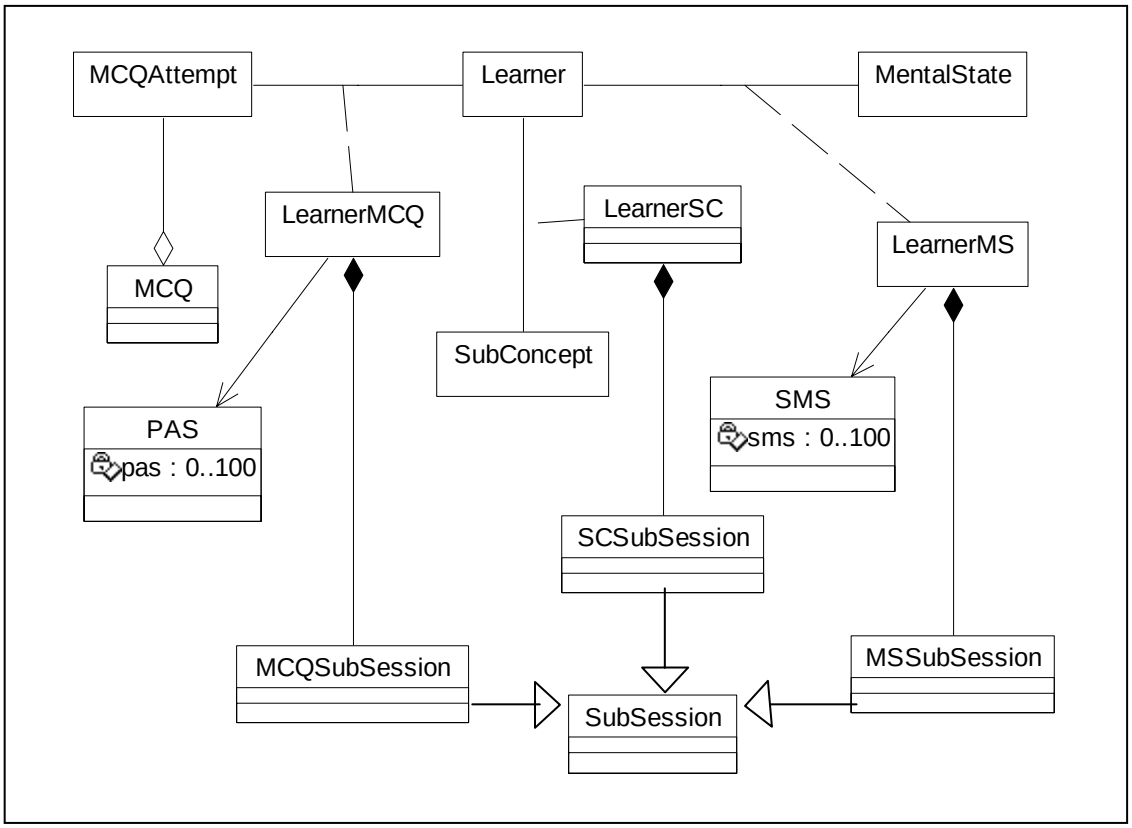


Figure 7.10 Class Diagram for Learner Model – Learner, SMS and PAS

7.7 Fuzzy Model for Uncertainty Handling

For each learner, the system keeps two measures: Global Learning Ability (GLA) and Strength of Mental State (SMS). The first measure, GLA, is an attribute of the learner that will change during their learning process. The second measure, SMS, is an attribute of both learner and a particular scaffolding stage. It will be updated more often than the GLA. Three linguistic terms High, Medium, and Low are associated with both measures.

Naturally, these values cannot be strictly defined. In order to handle uncertainty, both measures are considered fuzzy. After a formative assessment, the Mentor model needs to select pedagogical actions such as suitable feedback and next learning topic. A set of fuzzy rules (see Table 6.2) are designed to handle uncertainty associated with the selection process. A fuzzy variable called PAS is introduced in this research by combining the potential curriculum and feedback options. There are five ranks (levels 1 to 5) created, and these levels are considered linguistic variables associated with PAS. Figure 7.11 shows the hierarchical organization of the fuzzy variables used in this research.

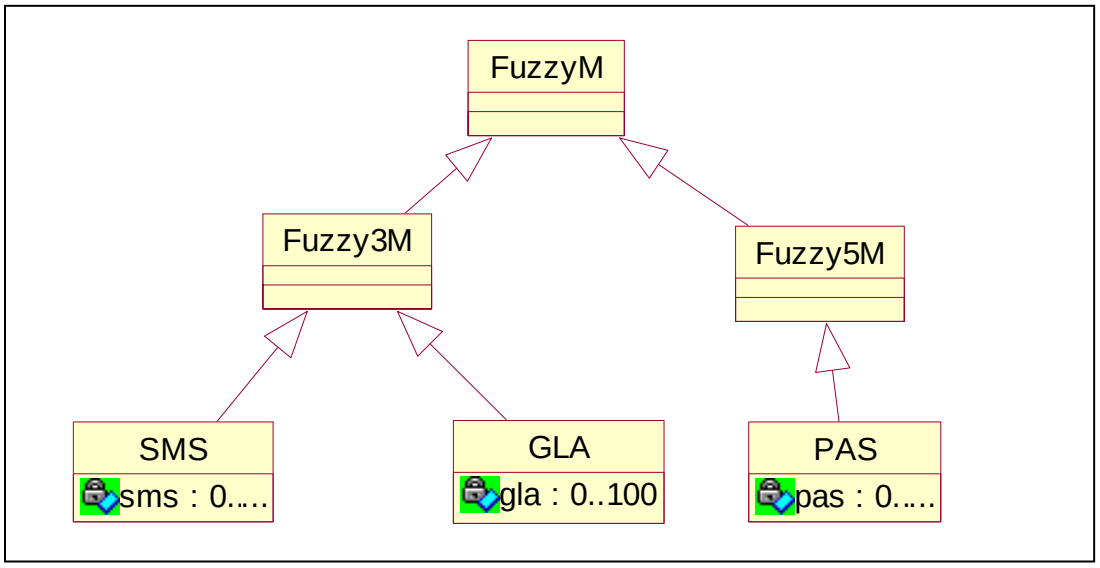


Figure 7.11 Uncertainty Handling – Fuzzy Model

The fuzzy measures SMS and GLA have three linguistic variables and identical membership functions. Therefore, an abstract class Fuzzy3M is created to generalize the associated classes. The fuzzy variable that represents the difficulty levels of MC tests also has a similar membership function (see Chapter 6). Therefore, it can also be inherited from the abstract class Fuzzy3M. Figure 7.12 includes a code segment for the fuzzification process based on the membership function in Figure 6.2. Moreover, Figure 7.13 gives a code segment for the fuzzy rule application process based on the rules specified in Table 6.2. The whole program for the prototype is given in Appendix C.

```
public static FuzzyMem3L fuzzify(short x){
   FuzzyMem3L f=new FuzzyMem3L(); // all levels are zero
   f.crisp=x;
   if ((x>=0) && (x<21))      f.level[0]=100;
   else if (x<41)  {  f.level[0]= (short)((40-x)*5);
                        f.level[1]=(short) (100-f.level[0]);
        }else if (x<61)   f.level[1]=100;
              else   if (x<81) { f.level[1]=(short)((80-x)*5);
                                 f.level[2]= (short)(100-f.level[1]);
                    }else  if (x<101)   f.level[2]=100;
     return f;
  } }
```

Figure 7.12 Code for a Fuzzification Process

```
public static FuzzyMem5L applyRule(FuzzyMem3L SMS, FuzzyMem3L D,
                    boolean correct) {
   if correct = TRUE
         applyRule (SMS, D, mxC); ; // rule matrix in Table 6.2
   else
         applyRule(SMS,D,mxW); ; // rule matrix in Table 6.2
end ApplyRule

public static FuzzyMem5L applyRule(FuzzyMem3L SMS, FuzzyMem3L D,
                    matrix mx) {
    short[ ] l = new short[5];    short minP;
    for (short i=0;i<5;i++)      l[i]=0;  // initially all levels are zero
    for (short i=0; i<3; i++)
        for (short j=0; j<3; j++){
              minP= (short) Util.U.min(SMS.level(i),D.level(j));
               if (minP > l[mx[i][j]-1])    l[mx[i][j]-1]= minP;
      }
   return new FuzzyMem5L( l );
```

Figure 7.13 Code for Fuzzy rule application process

7.8 Mentor Model Design

The Mentor model plays a central role in the prototype. Basically it controls and coordinates the key pedagogical functionalities of the prototype. It does not hold many data elements. The Mentor model provides appropriate service tailored to the learners whenever they ask for it. It will not provide pro-active assistance. The key responsibilities of the Mentor model are to perform two important pedagogical actions: curriculum sequencing and feedback. It may suggest the appropriate learning topic, suitable assessment, or relevant feedback in a situation where a learner is ready to move on (in the prototype, the potential situations include: when learners start their lesson, when they have completed the current material presented, after they provide answer for an MC test, and when they have learned the given feedback. The full-fledged curriculum sequencing task is out of the scope of the prototype, and in this research, it is limited to selecting appropriate scaffolding level (or in other words suitable assessment).

The sole class in the Mentor model is Mentor. It is not persistent. The Mentor class will be created afresh for each learning session. The control classes, main lesson control, feedback control, and MC test control are heavily associated with the Mentor model. The sequence diagram connected with feedback and assessment selection is given in Figures 7.14a and 7.14b.

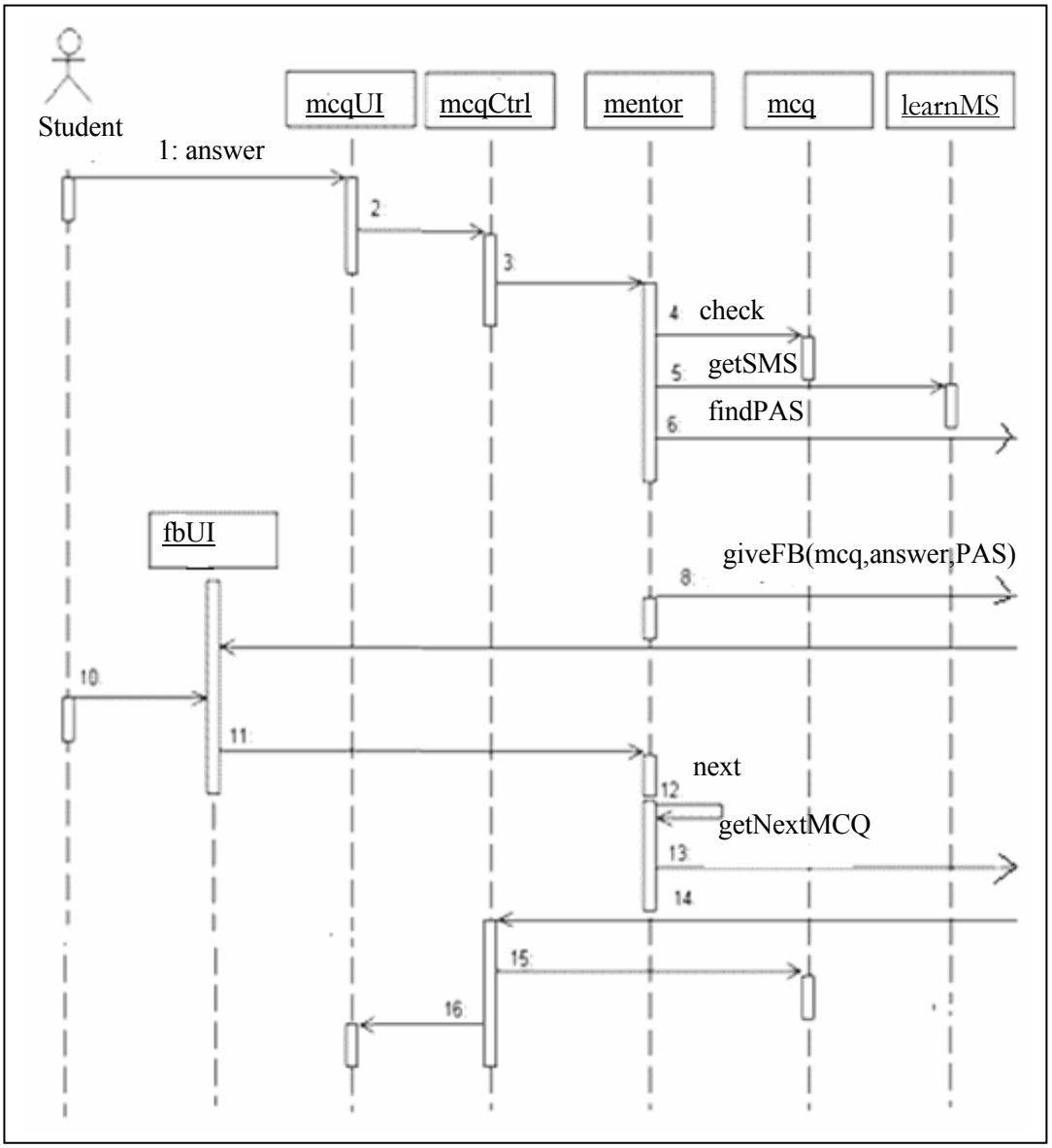


Figure 7.14a Sequence Diagram for finding PAS (continues in Figure 7.14b)

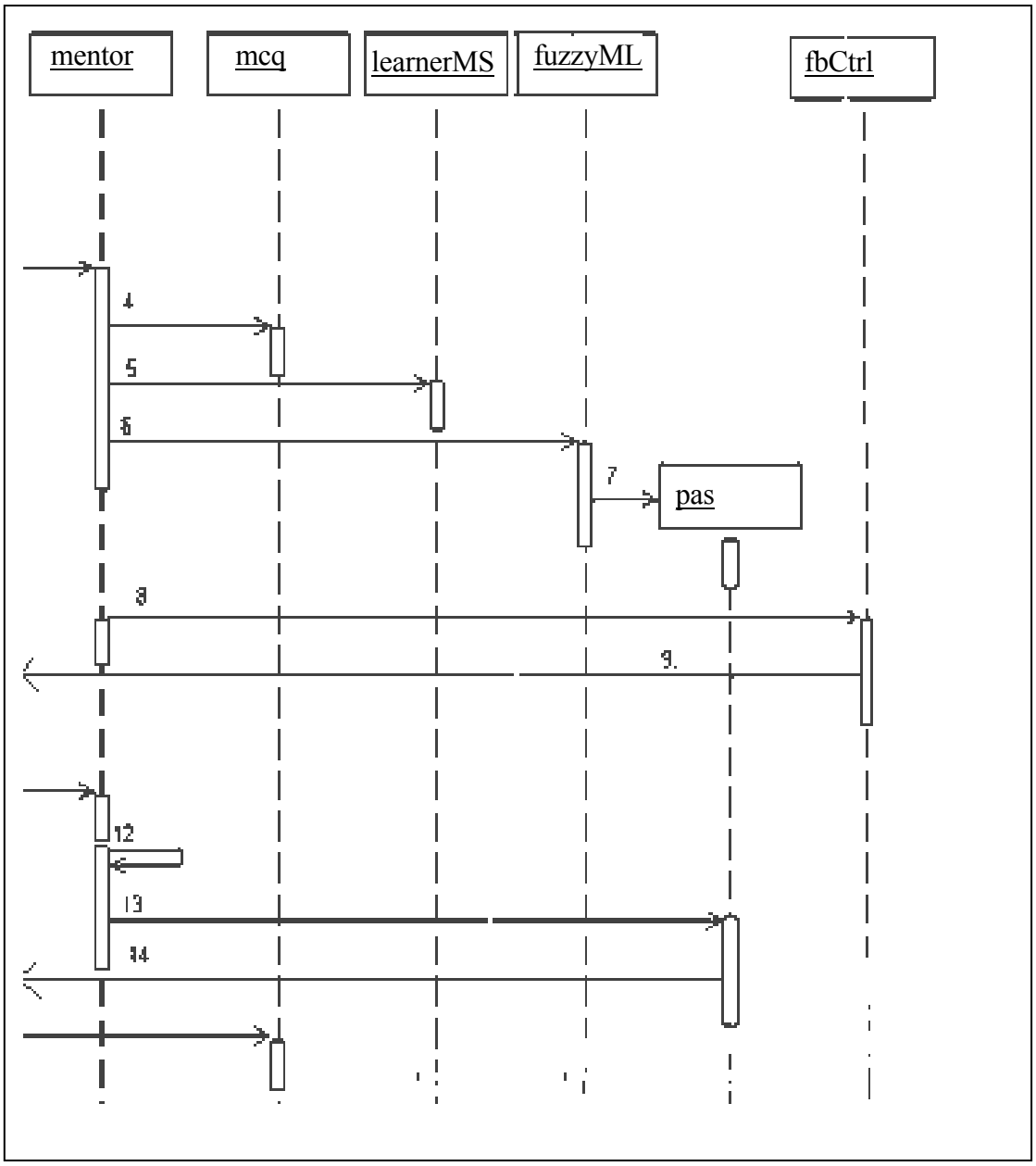


Figure 7.14b Sequence Diagram for finding PAS (continued from Fig 7.14a)

7.9 Database Design

All the classes associated with the domain model are persistent. Moreover, the important classes in the Learner model are also persistent. As mentioned before, JDatastore is used for database manipulation. The persistent classes are transformed to relational tables (Bennette *et al*, 2006). Firstly, the associations are designed (Figure 7.15), and then the inheritance hierarchies are flattened to association relationships. A high-level Entity

Relationship diagram is drawn (Figure 7.16). Thereafter, a set of relational tables are designed to satisfy at least third normal form (Figure 7.17a & b).

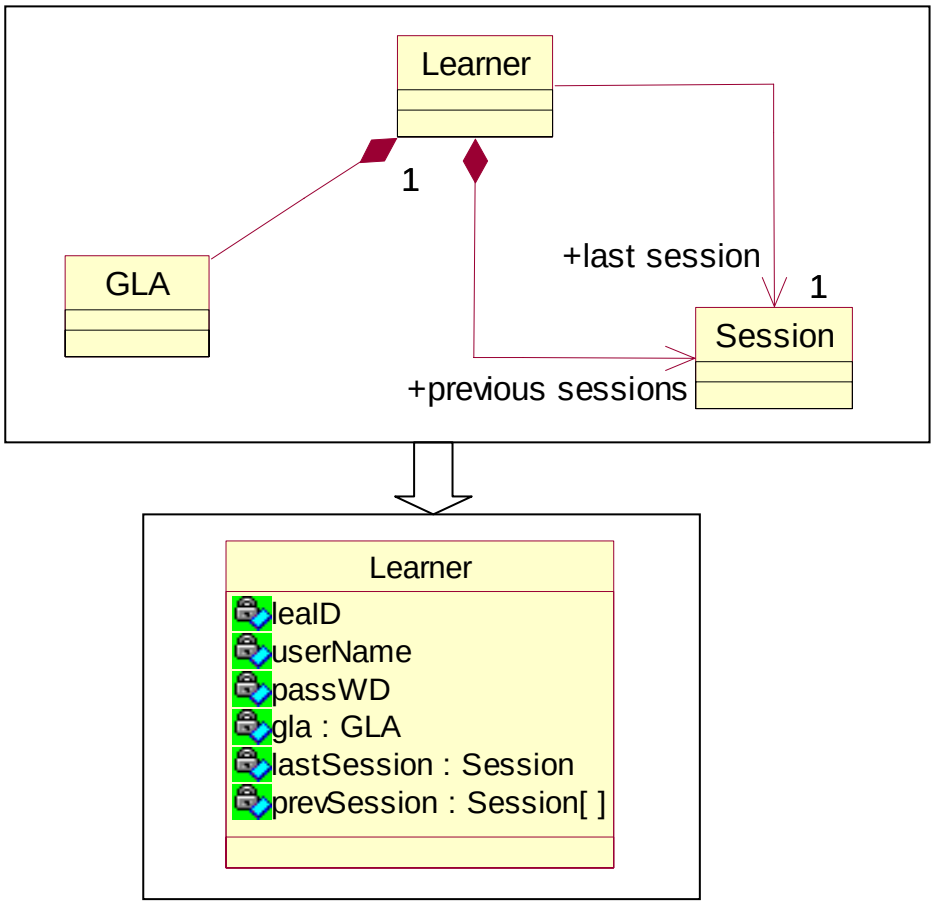


Figure 7.15 Designing Associations

The relevant physical tables were created using JDatastore Explorer, the front end of JDatastore in JBuilder environment (Figure 7.18). In order to simplify the database, slight modifications have been made in the class diagram. The database proxy pattern is used in this prototype (Bennette *et al*, 2006). However, instead of six different caches only one cache is used. An abstract data base broker class (called LozDB) is used as the parent class for all other database broker classes. The code segments in Figure 7.19a, b & c illustrate the implementation of the abstract parent class, a database broker and the cache access respectively.

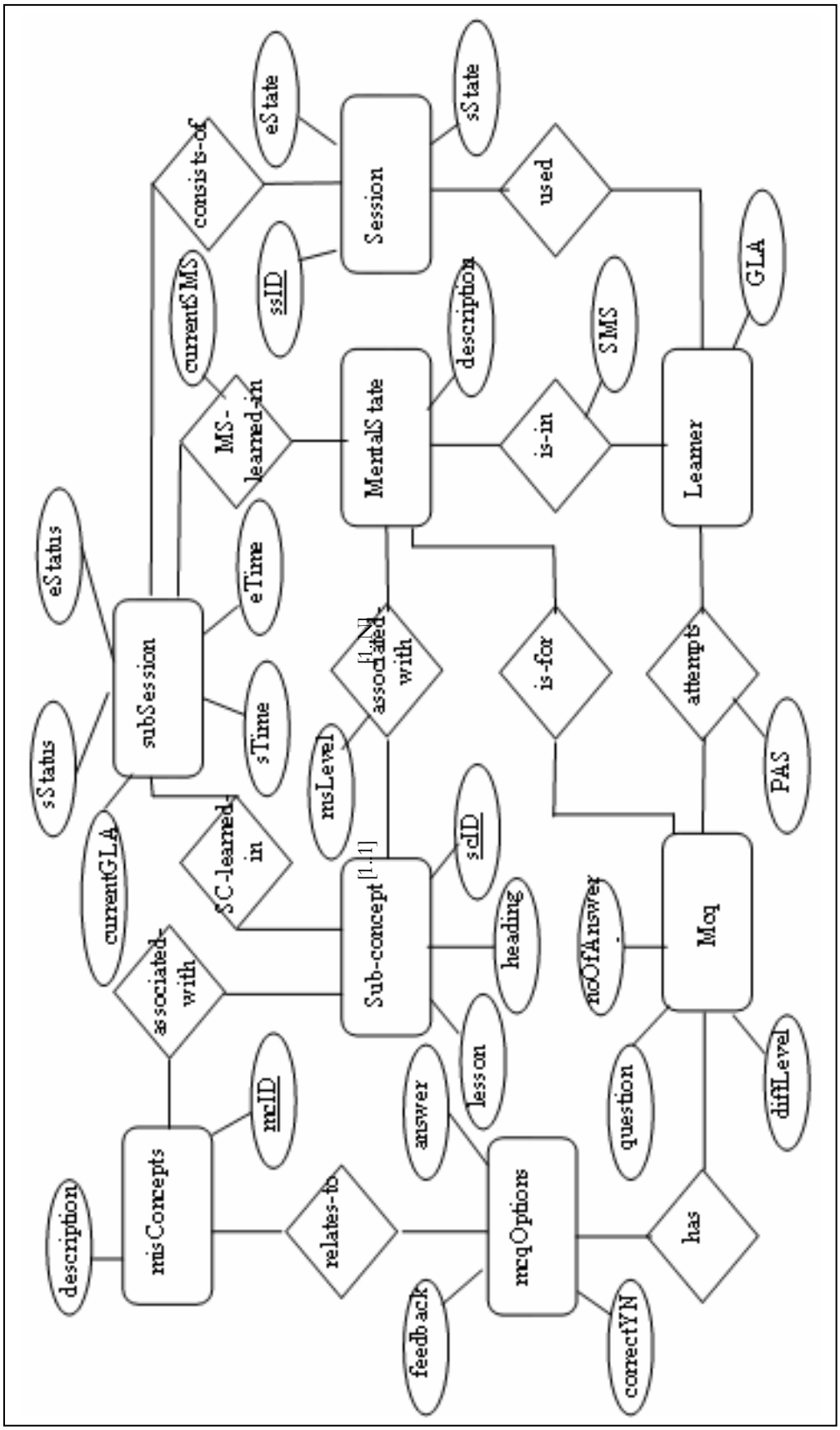


Figure 7.16 A high-level ER Diagram

Concepts & Assessment (see Figure 7.7)

- basicConcept (bcID, heading, description, #ofPoints, minT, maxT, ---)
- points(bcID, pID, pTxt, pBullet, PTxt, pSnd)
- subConcept(bcID, scID, notation, example, #ofMentalStates)
- additionalPoint(addPoiID, description, image) // for notation and example
- mentalState(msID, description, dependMSs, #ofMCQs)
- MSsINsubCpt (scID, msID, order)
- mainConcept(mcID, preCptID, #ofExplrCpt)
- subCptsINmainCpt(mcID, scID, order)
- misconception(misCptID, description, subCptID)
- mcq(mcqID, question, #ofCorrectAnswers, #ofDistractors)
- mcqAnswer(mcqID, mcqOpID, whyItIsCorrect)
- mcqDistractor(mcqID, mcqOpID, misConceptID, feedBack1, feedBack2);
- mcqOption(mcqOpID, description, image)
- mentalStateMCQs (msID, mcqID, difficultLevel)

Figure 7.17a Relational Tables – Concepts

Learner (see Figures 7.9 and 7.10)

- learner (leaID, leaDetailID, username, passWD, GLA, lastSessID)
- learnerDetail (ldID, leaID, prerequisites, preferences)
- learnerPrevSession(leaID, prevSessID)
- session (sessID, date, startTime, endTime, startState, endState)
- subSession(sessID, subSessID, type, sTime, eTime, sStatus, eStatus)
- mcSubSession(mcSsID, sessID, subSessID, startGLA, endGLA)
- learnerMC(leaID, mcID, currentGLA, currentState)
- leaMCSubSess (leaId, mcID, mcSsID)
- learnerSC(leaID, scID, currentGLA, currentSMS, currentState)
- leaSCSubSess (leaId, mcID, scID, subSessID)
- learnerMS(leaID, msID, currentSMS, currentState)
- leaMSSubSess (leaId, msID, startSMS, endSMS, subSessID)
- learnerMCQ(leaID, mcqID, attempt, startSMS, currentState)
- leaMCQSubSess (leaId, mcqID, attempt, PAS, subSessID)

Figure 7.17b Relational Tables - Learner

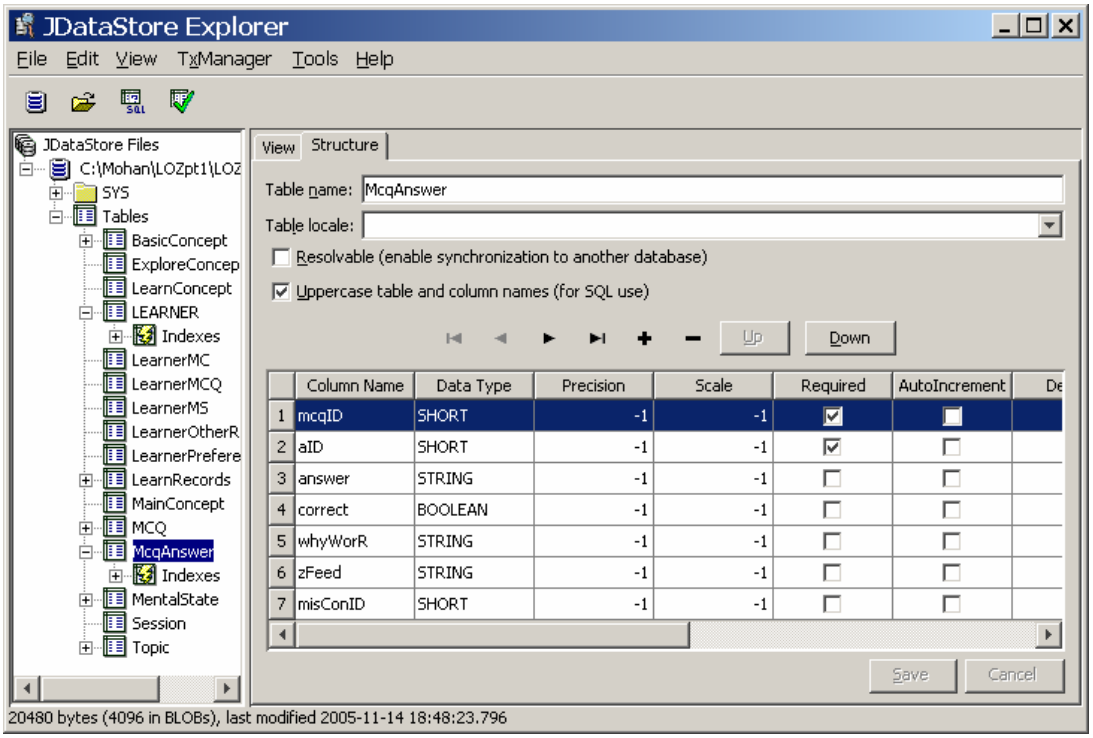


Figure 7.18 Offline Table Definitions

```
public class LozDB {
    protected Database LOZdB = new Database();
    protected ConnectionDescriptor connD = new ConnectionDescriptor
    ("jdbc:borland:dslocal:"  + "C:\\Mohan\\LOZpt1\\LOZdB\\LOZdB.jds",
                    "sm", "tm", false, "com.borland.datastore.jdbc.DataStoreDriver");
    public LozDB() {
          try {  jbInit();
         }   catch(Exception e) (Dix et al. 2007)
      }
    private void jbInit() throws Exception {
     LOZdB.setConnection(connD);  }
}
```

Figure 7.19a Code for the Abstract Parent class of Database Brokers

The singleton pattern is used whenever necessary (for example, the database broker class BasicConceptDB and the instance bcDB in Figure 7.19b).

```
public class BasicConceptDB extends LozDB{
 private static BasicConceptDB bcDB = new BasicConceptDB();
 public BasicConceptDB(){  }
 public static BasicConceptDB bcDB(  ) {
    return bcDB;  }
 public BasicConcept findaBasicConcept(short bcID){
  short type=1;
  BasicConcept bc = (BasicConcept)Cache.find(new Short(bcID),type);
  if (bc==null){      bc=findINdb(bcID);
        Cache.put(new Short(bcID), bc,type);      }
  return bc;
 }
 public BasicConcept findINdb(short bcID){
  QueryDataSet getBasQ = new QueryDataSet();
  QueryDescriptor qD = new QueryDescriptor(LOZdB, "SELECT \"BasicConcept\".*"+
    " FROM \"BasicConcept\" WHERE \"BasicConcept\".\"bcID\" = " + bcID ,
                              null, false,   Load.ALL);
  getBasQ.setQuery(qD);     getBasQ.executeQuery();    LOZdB.closeConnection();
  if (getBasQ.isEmpty())
   return null;
  BasicConcept bc= new BasicConcept( bcID, etBasQ.getShort(2),getBasQ.getString(3),
       getBasQ.getString(4),getBasQ.getString(5),getBasQ.getString(6),
       getBasQ.getShort(7), getBasQ.getShort(8), getBasQ.getShort(9));
   getBasQ.closeStatement();
 return bc;
  }
`
```

Figure 7.19b Code for a Database Broker

```
public static Object find(Object key, short type ){
    Map map=findmap(type);
    return map.get(key);
 }
 public static void put(Object key, Object value, short type ){
    Map map=findmap(type);
    map.put(key,value);
 }
```

Figure 7.19c Code for the Cache Access

7.10 Interface Design

Generally, the interface is an important component of interactive learning systems. However, as mentioned earlier, the key purpose of this prototype was to evaluate certain functional properties of the system, and the interface of this prototype was not a major consideration. Nevertheless, in order to avoid adverse effects on the evaluation process, a limited interface design is performed.

Dix *et al* (2007) describe some guidelines and rules for user interface design. First of all, the user profile is created. Relevant PERSONA (*ibid*.) and typical scenarios are described. The main group of potential users of the proposed system are the third year undergraduate or graduate students. Software practitioners can also use it for their professional development. As a starting point, a PERSONA and two typical scenarios are specified (Figure 7.20 a, b & c).

> Sam is a computer science undergraduate student. He is considered a cool-guy by his mates. He is currently in the third year. He has already studied the basics of software engineering and knows object oriented concepts and Z notation. He is not a straight A student, but his grades are fairly good. Particularly, Sam likes Z notation. His mathematical knowledge is excellent. Sam is currently studying an advanced software engineering course, which includes Object-Z notation. For this course, in addition to the usual learning material, an interactive learning system (LOZ) is also given on CD. Since Sam is also working as a part time employee at a popular supermarket, he mainly depends on this software to learn object-Z.

Figure 7.20a A PERSONA (Sam)

Based on the scenarios, task analysis has been conducted. The screen navigation design is then completed (Figure 7.21). The arrows indicate the possible navigational directions. For example, from the contents page one can go to the start of a lesson, whereas the contents page may be accessed from any part of a lesson during a learning session. From the Summary page one can directly go to any page of a lesson where the learner had left the session after his/her last use. Since the interface is implemented in JBuilder, screens can be simply created using built-in Swing widgets. Swing containers are used for creating sub-screens. The main screen is divided into five portions: menu bar, top tool bar,

main board, bottom tool bar, and status bar (Figure 7.6). Swing provides many built-in widget components.

During Screen design, Shneiderman's (2004) golden rules are closely followed. In particular, he advises to maintain consistency in action sequences, layouts and commands. He also advises to reduce short-term memory load of the users by keeping displays simple. In this design, grouping of widgets (interface elements such as buttons) are determined according to their semantics and order of usage. A balance is maintained between space utilization and context overloading. Similar widgets are grouped together. Symmetry of the related widgets is also checked. Affordability is checked for each of the widgets used. The ''New Learner' screen is given in Figure 7.22. More screen shots are given in Appendix B.

Sam is a new user, and he is asked to register first. He gives a user name and password. He is also asked to give estimations for his mathematical, Z and OO knowledge levels. After that, he views a page that describes the important features of the system. Then a content page is displayed with the first lesson high-lighted. Sam selected the lesson "Visibility List''. First, the precondition lesson (the UML accessibility options) is displayed on the main window. Since Sam is comfortable with the contents of the precondition screen he proceeds to the main lesson.

After learning the main lesson, while doing MC tests Sam feels that he gets unnecessarily detailed feedback. Sam knows that his current SMS score could play an important role in feedback selection; and therefore, he inspects the Learner model. He finds that the system's initial SMS value is very low. He changed it to a higher value. Gradually Sam builds his own understanding on the current lesson 'visibility list''. Finally, at the exploration stage, he is confronted with many challenging issues that contradict or further support what he has just learned. After some time he exits the system.

Figure 7.20b A Scenario – New Learner

Sam logs into the system. A brief summary of his last learning activity is displayed. After reading the summary, he closes the window. He is asked whether he wish to continue from his last session. He selects 'yes'. Before presenting the next lesson on Object-Z notation, initially, some relevant information on UML notation is displayed. After learning it, Sam presses 'continue'. The required lesson is given next. After learning, Sam presses 'continue' again, and this time a MC test is given. In the test, for a given description of a requirement, Sam is asked to identify the correct Object-Z specification among four answer options. Since it is the first level test, the relevant UML models are given as hints, and therefore, he only needs to transform UML models to Object-Z notation. Sam answers it correctly, and the system congratulates him. In addition, the system also explains why the selected answer is correct.

After confirming that his answer is correct Sam closes the feedback window. Another MC test is selected by the system. This time the test includes only partial UML support. Sam answers this question also correctly. System just confirms that his answer is correct. After that, another MC test is given. This time, totally no UML help is given. Therefore, Sam has to abstract necessary detail from the textual description in order to identify the correct answer. Sam knows that two options are not correct, and he guesses one of the other two. Unfortunately, it is not correct and appropriate feedback is given. After Sam read it, he understands the concept. The feedback is appropriate and sufficient. Finally, the system gave another MC test, but now at the same level. Sam answered it correctly. After confirming, system asks whether he want to proceeds to the next lesson. Sam selected 'no, and the system saves relevant detail before closing down.

Figure 7.20c A Happy day Scenario – Returning Learner

Dix *et al* (2006) argue that multi-modal presentation will be more effective than a single modal one. The learning materials are presented as bulleted abstracts and detailed explanation is given in voice. It may be noted that the facilities for sound editing are not sufficient for prototyping in JBuilder, and therefore, synchronizing bulleted points and voice explanation is very difficult. Moreover, MC tests are displayed on the 'main board', but the answering interface is given separately as a modal window (Figure 7.8). This arrangement will be more suitable for confidence based MC testing process.

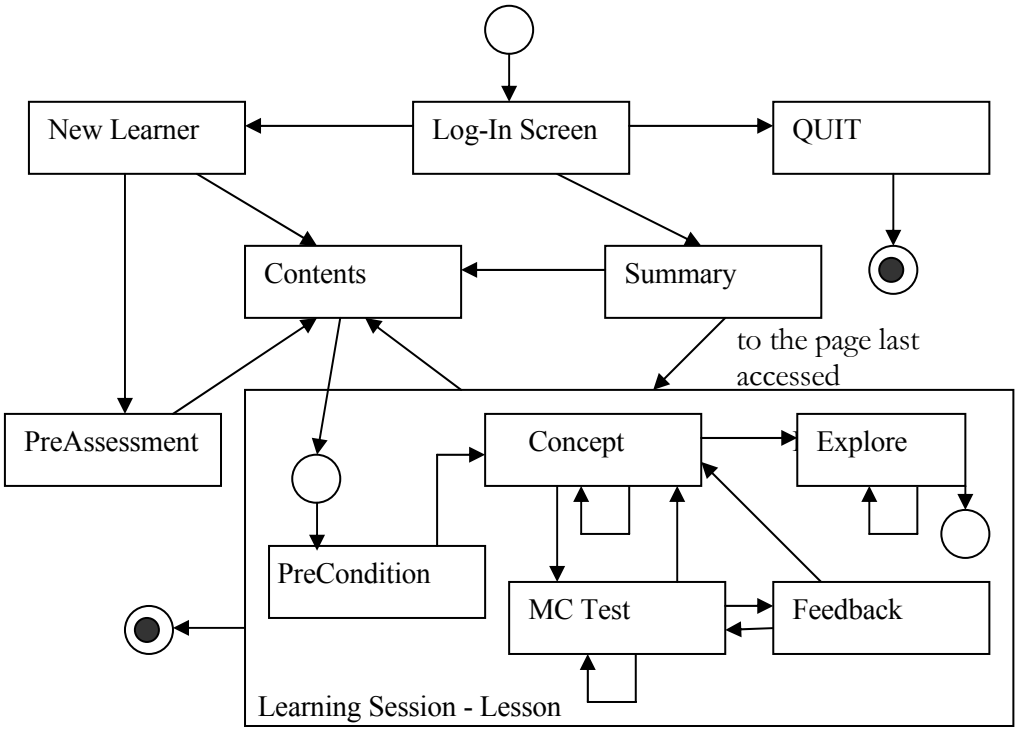


Figure 7.21 Screen Navigation

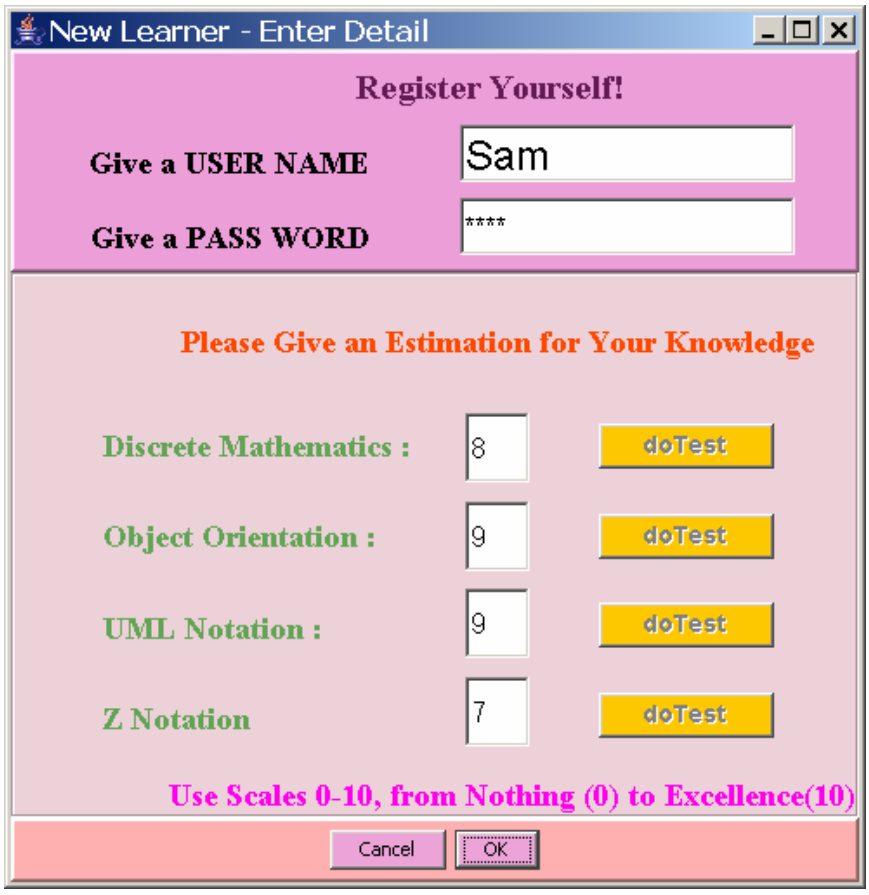


Figure 7.22 New Learner Registration Screen

7.11 Opening the Learner Model

Finally, the prototype was extended to include features related to opening the Learner model. Figure 7.23 shows the interface to display the current estimates of the Learner model in different views. This feature is not fully implemented- though the interface allows learners to modify their own estimate, the original model remain the same. However, the system may explain its behaviour based on the current Learner model estimates. Since the linguistic variables in a fuzzy model reflects real world concepts, and the underlying fuzzy rules resemble typical teachers' decision making processes, it is easy to explain its reasoning mechanism in every-day language (Figure 7.24).

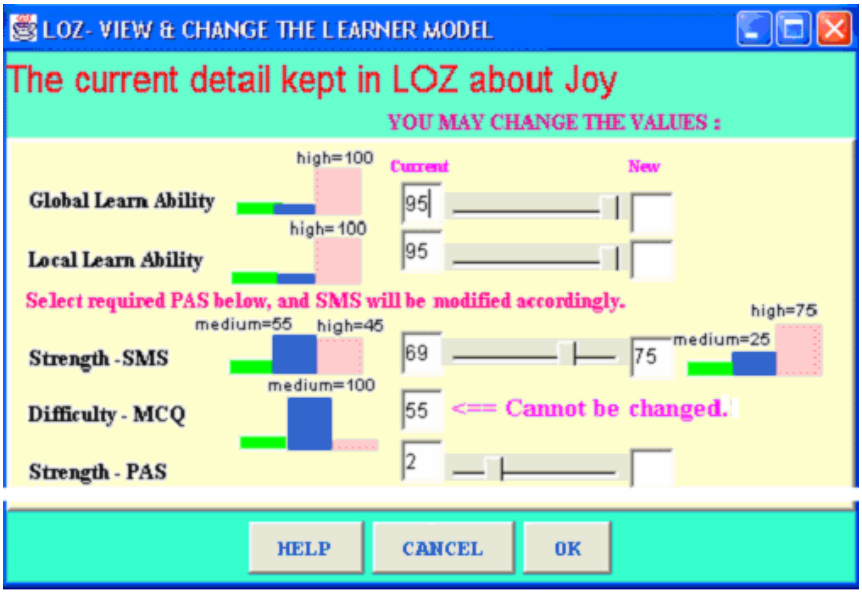


Figure 7.23 Learner Model Estimates

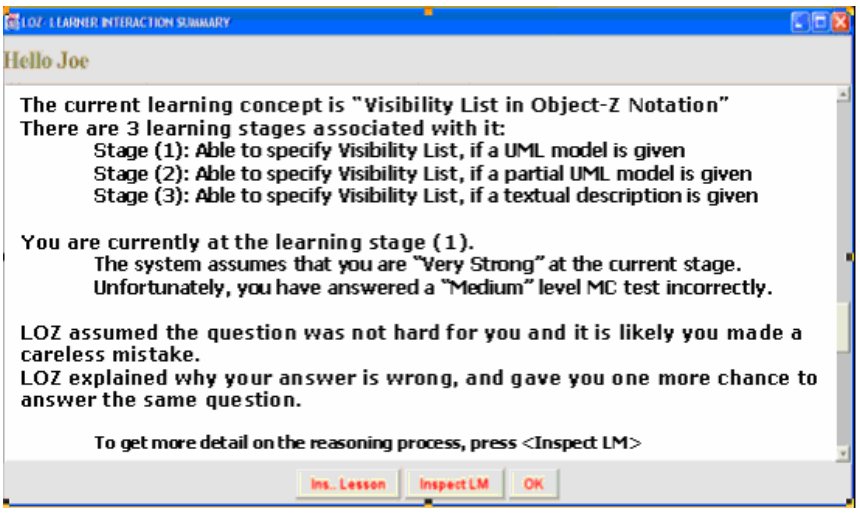


Figure 7.24 Explaining Learner Model Behaviour

7.12 Summary

A prototype has been developed to evaluate the thesis proposals. It can also be used to demonstrate that the proposed solutions are technically feasible. The prototype is created in an evolutionary fashion with reusability in mind. First of all, the scope of the prototype has been defined. A layered architecture has been adopted as a basic building structure. Initially, interface, control, business, and database layers are included. Clear distinctions are made between the tasks of classes of different layers. For example, only a class in the database layer can access the physical database. JBuilder is used as the programming environment, and JDataStore is used as the database environment. Changing the database would require only some of the classes of the database layer to change.

The following three main sub systems are identified: Learner model, Domain model and Mentor model. The Learner model includes personal and nearly permanent data about learners and other historical records of their past learning activities. It also includes a fuzzy model that could recommend suitable pedagogical decisions in uncertain situations. The Domain model includes not only the course material but also relevant assessments and feedback. The Mentor model, being the sole teaching unit, controls and coordinates the whole processes. An appropriate interface for supporting learning has been developed based on typical scenarios. Screen navigation has been considered. In the next chapter, the evaluation of the prototype is described.

Chapter 8

Evaluation and Discussion

8.1 Introduction

The issues related to the prototype development were discussed in the previous chapter. This chapter describes the evaluation of the prototype and discusses the results. The prototype was, in particular, used to examine the overall effectiveness of the CBL system (LOZ) designed in this research. More precisely, it was used to evaluate the key constituents (primarily Learner Model) and the underlying instructional strategies (FoPSI model) of the proposed system.

Section 8.2 discusses a set of research claims that are used as the basis for evaluation process in this research. Section 8.3 discusses both the subjective and objective evaluation strategies used in this research (such as experimental design, log files, and Questionnaires. Objective measures are used to evaluate the two key hypotheses. In order to independently evaluate the accuracy of the Learner model, a new fuzzy rule based prediction mechanism is included in the system. The system's predictions about a learner's performance and the learner's actual performance are recorded on log files. A questionnaire is also used to evaluate all the research claims subjectively.

Section 8.4 outlines the pilot study conducted to test the evaluation process itself.. The evaluation of the two key hypotheses is discussed in Section 8.5. The results and discussions of all the tests (related to all the claims) are included in Section 8.6. Finally a combined summary based on both objective and subjective evaluations is provided.

8.2 Research Claims

As described earlier, LOZ is not intended to replace the human teacher; instead, it is proposed to be used as a revision tool. LOZ incorporates a Refinement unit; uses problem transformation based scaffolding strategy for teaching, and includes a Learner model to support adaptable scaffolding.

There are five claims made in this research. The first two of them are the kernel of this research and are given as the following hypotheses.

Hypothesis 1: In general, the overall strategy proposed for designing CBL systems for learning complex domains using problem transformation technique is effective. In particular, the CBL system LOZ is suitable for learning the exemplar domain- Object-Z notation.

The Learner model plays vital role in LOZ. The second hypothesis is about the effectiveness of the Learner model.

Hypothesis 2: The proposed fuzzy logic based Learner model for adaptable scaffolding models the learner sufficiently.

It may be noted that the effectiveness of the Learner model will positively affect the effectiveness of the whole CBL system. Therefore, the first hypothesis includes the second one to some extent. However, since the Learner model design is a key proposal of this research, the second hypothesis is formulated as one of its own right.

Other than the above two hypotheses, there are some other issues, though less important, which need to be evaluated, particularly, the FoPSI model. This instructional model consists of many features such as pre conditioning, exploring etc. Although, the first hypothesis includes all of these issues, they can also be tested separately.

Claim 3: The FoPSI model in LOZ is efficient for instruction (especially for learning Object-Z using adaptable scaffolding).

Opening the Learner model gives many advantages for learning. A strategy for externalizing the fuzzy based Learner model (and the Mentor model that uses scaffolding strategy) is proposed in this research.

Claim 4: Externalizing the fuzzy Learner model is effective.

Finally, a confidence based multiple choice schema is designed in this research to support formative assessment. This scheme enables the system to provide appropriate feedback essential for adaptable scaffolding strategy.

Claim 5: The CBM schema measures the learners' state of knowledge precisely (compared to the traditional MC test schemas).

Evaluating the last three claims is out of the scope of this research. However, some pilot testing information on these claims (based on subjective evaluations) is included in this chapter for completeness.

8.3 Evaluation Strategies

Evaluations involving well designed experiments and rigorous statistical analysis are very common in general education research (Ravid 2000; Gorard 2001; Coladarci *et al.* 2004). In contrast, serious evaluations are rarely reported in the CBL systems research literature. Nevertheless, Chin (2001) listed nine studies (from the nineteen nineties) that reported empirical evaluations on Learner models.

In this research, the first hypothesis was evaluated objectively based on quantitative data obtained from an empirical experiment. Section 8.3.1 gives the background information on the evaluation process including the conditions of the experiment. The second hypothesis was also evaluated objectively using quantitative data, but directly stored in relevant log files during the experiment. The estimates of the Learner model at the key interaction moments are recorded, and later, these records were compared with learners' actual achievements. However, in order to create appropriate logs the Learner model was slightly modified. Section 8.3.2 explains this issue in more detail.

In order to evaluate the rest of the claims, a questionnaire was administered to perform subjective evaluation. The questionnaire also includes some items related to the first two hypotheses. Section 8.3.3 discusses the questionnaire design in detail. Finally, the claim about the CBM method could also be evaluated objectively. In addition to the traditional MC test scheme, a new MC test answering interface and marking schema based on CBM method is also used during the experiment. Section 8.3.4 discusses this matter in detail.

8.3.1 Evaluating the System: Empirical Study

Different type of experiments may be designed to test the hypotheses formulated in the previous section. In particular, for evaluations related to education, Ravid (2000) classifies the experimental group designs into three categories: pre experimental (no control group/ or no pre-test), quasi-experimental (two groups, control and treatment; but, both groups may not be similar), and true experimental. Particularly, with pre-experimental design, evaluations may be conducted using two groups (without pre-tests where groups may not be similar) or a single group (with pre-test). In this design, the effect of the treatment is assessed by the difference between the subject's post-test and pre-test scores. To be statistically sound, the size of the group should be large enough. Although the size around 20 is preferable, in well-designed studies the size of 10 is acceptable (Ravid 2000 p. 47). If two groups are used, the control group may or may not receive a variant treatment, while the experiment group receive the treatment related to the hypothesis. For the evaluations involving two samples, one of the relevant Student-t tests may be used for statistical analysis. Before doing the actual experiment, a pilot study was conducted to help to identify potential problems in the planned procedures.

The following two experimental design options were considered to test the first and second hypothesis. The first option, true experimental design, is discussed below.

Select two groups of subjects, control group and treatment group.
Assign subjects randomly to the groups (or pair wise assign).
Give the same pre-test for both groups (a set of MC tests).
Give printed materials to the control group to learn a lesson in Object-Z
Meantime, give the current version of LOZ to the treatment group
Give the same post-test for both groups (another, but a set of similar MC tests).

The performance in the post-tests of both groups can be used to test the hypothesis. If the groups are randomly assigned, the samples are independent (Ravid 2000 p. 190), otherwise, if they are pair wise assigned, which is also known as matched-subject design (Coladarci *et al.* 2004), the samples are dependant. Let μ1 and μ2 be the means of the post-test scores of the control and treatment groups respectively. The null hypothesis holds that there is no statistically significant effect in using LOZ compared to learning using printed material. The alternate or test hypothesis holds that using LOZ to learn Object-Z has statistically significant effect over learning using printed material.

The related null hypothesis H0: $\mu_1 = \mu_2$

The alternative hypothesis H1: $\mu_1 < \mu_2$

If the groups are randomly assigned, the Student-t test for independent samples can be used to test the hypothesis. If they are pair wise assigned, Student-t test for pair wise samples need to be used.

To reduce the threats to the internal validity, the potential effects of extraneous variables need to be minimised. In this test, the control group and treatment group should be similar before treatment. The performance in the pre-tests of both groups can be used to ensure that this condition holds. In this case the alternative hypothesis should be nondirectional (H1: $\mu_1 \neq \mu_2$, where μ_1 and μ_2, are related to pre-tests). Another threat to the internal validity of the test is the effect of pre-test on the post-test scores. This is a common problem for all the evaluations of this kind. To minimize this effect, the answers of the pre-test should not be given to the subjects before the post-test.

The Learner model is the key component of LOZ; and therefore, to some extent, the first hypothesis covers the second hypothesis. However, to test the second hypothesis independently, a similar true experiment design may be used. But, instead of the plain printed learning material, the control group should be given a flat version of LOZ (Learner model may be disabled or it may give random feedback). This will ensure that the difference, if any, could be attributed to the Learner model and not to any other features of LOZ.

The second option was pre-experimental design (or repeated-measures design).

Select a group of subjects.

Give a pre-test (a set of MC tests).

Give the current version of LOZ to the group to learn a lesson in Object-Z

Give a post-test (another set of similar MC tests).

To minimise the potential threats to the internal validity of this test, the pre-test and post-test must be almost equally difficult. For this purpose, two different but similar tests (say T1 and T2) may be prepared, and then, half of the subjects (randomly assigned) may be given T1 for pre-test and T2 for the post-test. The order must be reversed for the other half. For them, T2 is given for the pre-test and T1 is given for the post-test. This arrangement ensures that none of the subject faces the same questions in both pre and post-tests, and also it reduces any potential adverse effects due to variation in difficulty levels of pre and post-tests. Similar to the true-experimental design discussed before, this pre-experimental design also has a threat to the internal validity due to the potential effect of pre-test on the post-test scores.

The semantics of the null and alternate hypotheses are slightly different from the true-experimental design. Let μ1 and μ2 be the means of the pre and post-test scores of the group respectively. The null hypothesis holds that there is no statistically significant effect in using LOZ for learning Object-Z. The alternate or test hypothesis holds that using LOZ to learn Object-Z has statistically significant effect.

The related null hypothesis H0: $\mu_1 = \mu_2$

The alternative hypothesis H1: $\mu_1 < \mu_2$

Since the samples are dependant and the scores are paired for each learner, Student-t test for pair wise samples was used to test the hypothesis.

Though it is not necessary, it is possible to ensure statistically that the both tests do not differ significantly. For this purpose, the performances of the pre-test and post-test scores of each learner were coupled, and Student-t test for pair wise samples is used again to test this claim. In this case, the alternative hypothesis should be non-directional (H1 : $\mu_1 \neq$

μ_2, where μ_1 and μ_2, are related to pre-tests). Another threat to the internal validity of the test is the effect of pre-test on the post-test scores. This is a common problem for all the tests of this kind. To minimize this effect, as stated before, the answers of the pre-test should not be given to the subjects before the post-test.

Nevertheless, there are many other potential threats to internal and external validity and all of them cannot be totally controlled (Chin 2001; Coladarci *et al.* 2004). Chin (2001) suggests some rules to minimize the effects of nuisance variables.

Although the first option (true experimental design) is efficient, for practical reasons, the pre-experimental design is used in this research. The subject domain of LOZ is an advanced discipline in computer science and taught normally for third or fourth year undergraduate students, and therefore, it is very difficult to find sufficient number of students to form two homogeneous groups of reasonable size. As a result, the repeated-measures design, which is based on only one group of subjects, is used here. This approach is common in empirical evaluations in educational research when the resources are limited (Coladarci *et al.* 2004).

8.3.2 Evaluating the Learner Model: Log Files

The second hypothesis claims that the Learner model of LOZ accurately models the learners. An accurate Learner model could provide personalized support to the learners effectively (such as appropriate feedback and curriculum sequencing). The accuracy of the Learner model may be tested indirectly using the pre-post-testing method described before. In addition to that, the accuracy of the Learner model may be evaluated directly by testing the association between the models of students and the students themselves. In statistical terms, the correlation between the expected performance of a learner based on the system's Learner model and their actual performance indicates the accuracy of a Learner model.

Testing the accuracy of a Learner model can be performed at two levels. Corbett *et al* (1993) refers these two levels as testing for external and internal validity of the Learner model. At the higher level, after the system has been used for learning, the correlation between the post-test performance and the states of the final Learner models may be used.

A high correlation indicates high accuracy. On the other hand, during the system in use, the learner needs to answer many MC tests. The system may predict the performance of a student based on their current state of the Learner model. To be accurate, the Learner model should make significantly higher number of correct predictions than wrong predictions. The related figures, the number of correct and incorrect predictions, for each learner can be considered as a repeated measure, and the Student-t test for dependant sample may be used to test this claim. To perform this test, performance of a student should be explicitly predicted by the system. Therefore, a performance prediction mechanism based on fuzzy logic is incorporated with the system. The next paragraph discusses this mechanism in detail.

Performance Prediction Mechanism

Firstly, we shall revisit the variable Performance (P) described in Chapter 5. If a confidence-based MC test is used, different levels of P can be identified (for example correct, incorrect and medium). As with the variables discussed in Chapter 5, the boundaries of the performance levels cannot be defined strictly. The fuzzy set membership function given in Figure 8.1 will be assumed for P. If only two linguistic values are assumed, the membership function may be the same as Figure 8.1, but with the medium part removed.

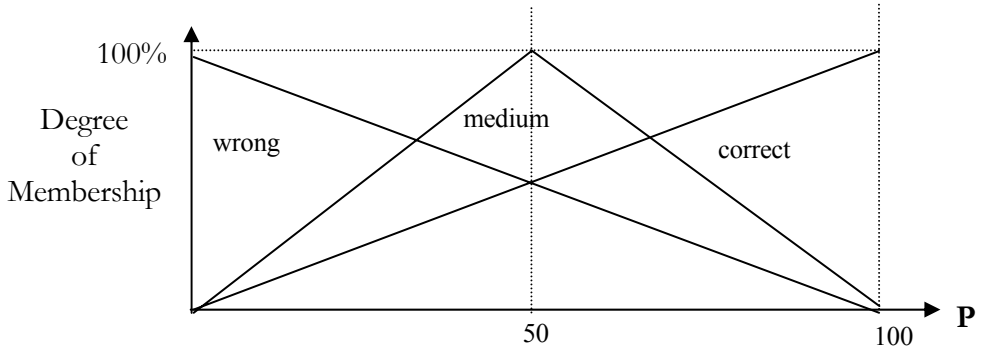


Figure 8.1 Fuzzy Membership Functions for Performance

As discussed in Chapter 5, P depends on SMS and D, and when P becomes available, the suitable pedagogical action depends on all the three variables; SMS, D and P (Figure 8.2) For evaluation purpose, the first set of causal relationships (darkened lines in Figure 8.2, see Figure 6.4 also) are considered important. For a particular learner, before answering

an MC test, the expected performance could be determined using the values SMS and D. The fuzzy rules that govern these casual relationships are given in Table 8.1.

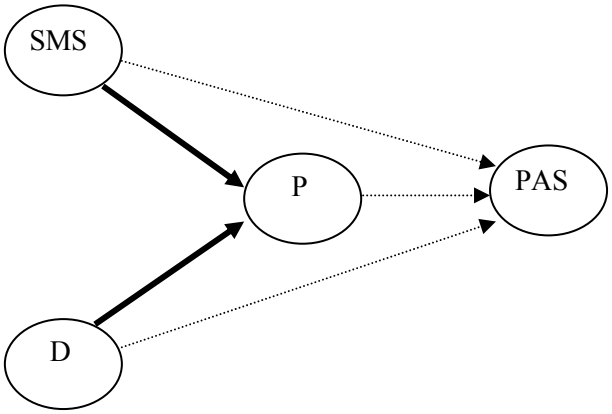


Figure 8.2 Causal Relations for PAS

Mental State (SMS)	Difficulty level (D)	Performance (P)
Strong	Low	Correct
Strong	Moderate	Correct
Strong	High	Medium
Medium	Low	Correct
Medium	Moderate	Medium
Medium	High	Medium
Weak	Low	Correct
Weak	Moderate	Medium
Weak	High	Incorrect

Table 8.1 Fuzzy Rules for Performance

As with the fuzzy rules defined in Table 6.2, the fuzzy rules in Table 8.1 closely reflects the decision making process of a typical human mentor in different situations. For example, human mentors usually expect that the experts will perform well on easy assessments. Similar to the Learner model mechanism described for Phase-II (described in Chapter 9), another set of fuzzy rules to determine PAS values based on predicted performance and actual performance (if both are fuzzy) may be defined (see Table 9.1). Instead of directly using Table 6.2, Tables 8.1 and 9.1 may be used together in order to select PAS.

In order to evaluate the accuracy of the Learner model, a log file is maintained in the system. The Learner model will predict P for a learner before a MC test is answered. These predictions and learners actual performance on each assessment are recorded. The

prediction is made based on the fuzzy-rules in Table 8.1, which uses the current strength of mental state (SMS) and the difficulty level (D). For example, if a learner with the Mental State (SMS=65) failed a question with difficulty level (D=35), the system expects medium level performance. In the traditional MC test scheme, it may be either correct or incorrect. The underlying fuzzy mechanism used for this example is illustrated in Appendix E.

8.3.3 Questionnaire

Questionnaires are well-established techniques for evaluating something subjectively based on the relevant people's opinion or some other related data (e.g. demographic data). Designing efficient questionnaires demands significant skills and effort (Berdie *et al.* 1986). A questionnaire is designed in this research to evaluate all the five claims including the first two hypotheses. In particular, for the last three claims this is the only source of evaluation reported in this dissertation.

The questionnaire consists of 13 statements in four parts. All the items in the questionnaire are close-ended, except the comment section. The statements in the questionnaire are designed to record the learner's perception of certain features of the system. The Likert style five point rating scale (ranging from strongly disagree to strongly agree) is used for measuring responses. Table 8.2 gives short descriptions of the statements in the questionnaire (for a full version, please refer to Appendix D).

The first part of the questionnaire is related to the FoPSI model and contains five items. Each item is related to one of the following proposed features; pre-conditioning, scaffolding, exploration, Learner model, and refinement. The second part of the questionnaire contains four items, and all of them are related to externalizing Learner models. The third part contains two items, and both are related to the CBM strategy. The fourth part includes two statements related to general issues such as user acceptance and satisfaction. It can be noted that the first and last parts of the questionnaire are related to the first hypothesis described before, and therefore complement the empirical evaluation strategy. Similarly, the fourth statement in the first part is related to the second hypothesis as it particularly refers to the Learner model of the system. In addition to close-ended

items, the subjects are also encouraged to write free form comments about their experience in using LOZ for learning.

No	Group	Related Issue	Statement (A short description)
1	FoPSI	Pre-conditoning Phase	Pre-conditioning phase is helpful
2	FoPSI	Scaffolding Strategy	Scaffolding is handled effectively
3	FoPSI	Exploration Phase	Exploration feature is useful
4	FoPSI	Feedback in Learner Model	System provides adaptable feedback
5	FoPSI	Refinement in Learner Model	Providing Java code motivates learning
6	Open LM	Opening Learner Model	Viewing System's estimate useful
7	Open LM	Modifying Learner Model	Facility for modifying the estimate is handy
8	Open LM	Modifying Learner Model	Feature to change the feedback contents is good
9	Open LM	Modifying Mentor Model	Facility to modify scaffolding steps is useful
10	CBM	MC test strategy	CMB allows to express partial knowledge
11	CBM	MC test strategy	CBM is better compared to traditional scheme
12	General	Learner Acceptance	System gives suitable learning environment
13	General	Learner Satisfaction	I will not hesitate to recommend it to others

Table 8.2 Subjective Evaluations – Questionnaire

8.3.4 Evaluating CBM Schema

One of the research claims is that the proposed CBM schema is suitable for adaptable scaffolding strategy. In other words, the CBM schema can identify the partial knowledge efficiently. In order to test this hypothesis, there are two types of answering methods included for each MC test item in both pre and post-tests. In addition to traditional method (selecting only the most suitable option), the students may also be asked to indicate their level of confidence on each answer option (Figure 8.3). Before that, the students should be given sufficient time to understand the format of the test and the marking mechanism.

First of all, the exams are to be reliable. The Cronbach's co-efficient alpha may be used for reliability analysis (Ravid 2000). Next, item analysis may be conducted to test whether

CBM measure the same student knowledge as traditional test does. The Pearson's co-efficient may be used for this purpose (Bradbard *et al.* 2004). Finally, the score on each test items may be compared to identify how far the proposed CBM strategy is efficient for determining partial knowledge (*ibid.*). Note that, the answering interface for providing confidence for options is slightly different from that was originally designed in Chapter 5 (Figure 5.6). This change was made based on the pilot study (will be discussed in the next section). However, the marking mechanism remained unchanged.

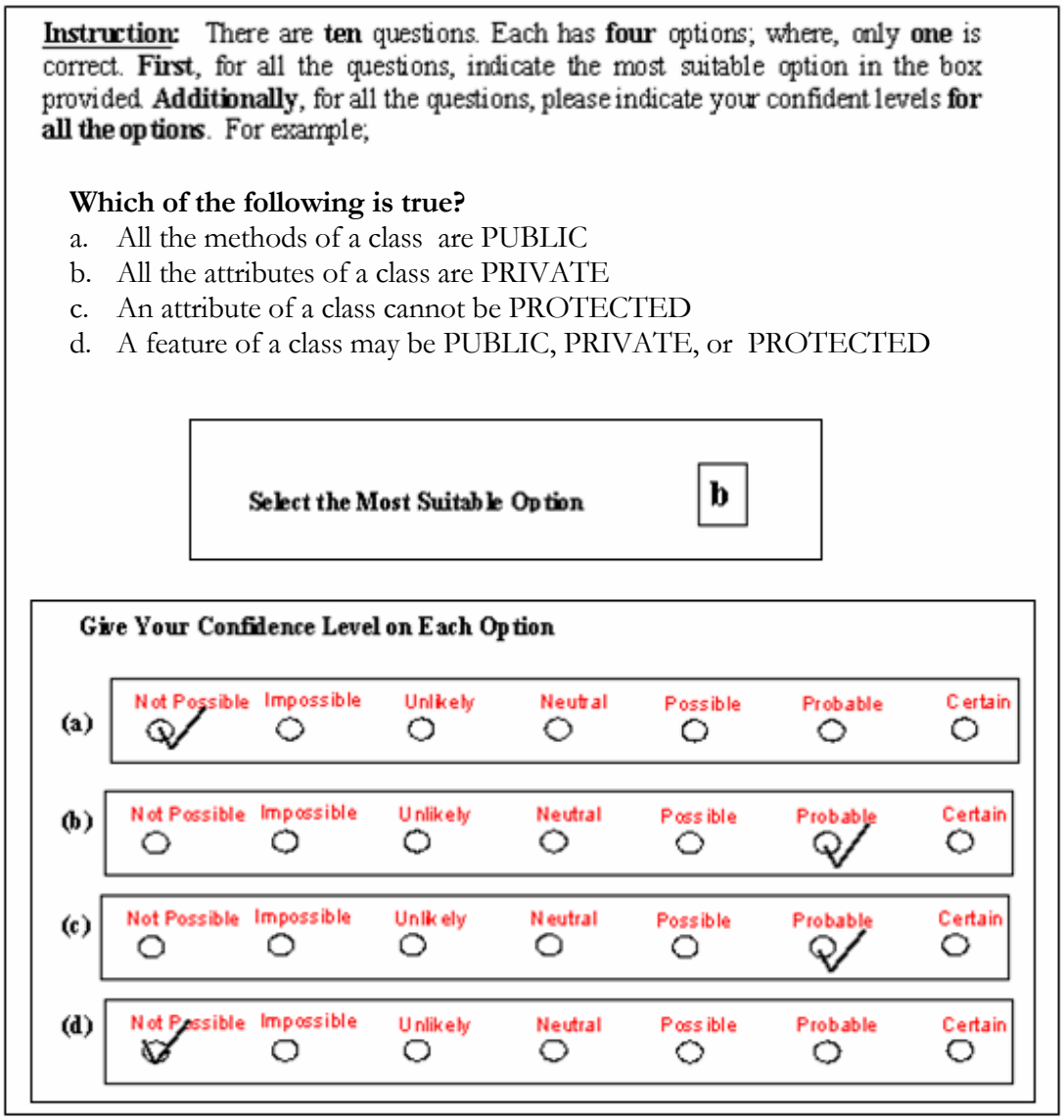


Figure 8.3 MC Test Answering Interface – CBM Strategy

8.4 Pilot Study

Before conducting the actual experiment a pilot study was conducted. A postgraduate student volunteered to conduct the pilot study. After the introduction and other formal procedures, a pre-test on an Object-Z concept (Visibility List) was given to the student. The pre-test consists of ten MC questions. Later she was asked to use the prototype to learn the same concept (Visibility List). One hour was allocated for this process. Finally, a post-test was given. The time allocated for pre and post-tests was forty minutes each. The pre and post-tests are carefully selected so that they are almost at the same difficulty level. After the post-test, a questionnaire was completed by the student.

Based on the pilot study, several changes were made to the prototype as well as to the evaluation procedures.

- The interface of LOZ was slightly changed.

  Sound feature and the buttons related to uncompleted features were disabled (e.g. Notebook).
- The time related predictions in the Learner model were disabled.

  The original system tracks the time spent by the learner on each stage, and use this information for various decisions. But, usually, students get distracted from learning, and the time estimate may become unreliable.
- The answering interface for the CBM strategy was changed.

  There were three separate spectra in the original interface: correct, incorrect, and not sure (Figure 5.6). Students are asked to provide their confidence through any one of the spectra. However, for marking purpose, all three spectra are considered continuous. During the pilot study, the student used only one spectrum for all the twenty questions. In order to reduce complexity, the interface is changed: it now includes only one continuous spectrum and the confidence levels are annotated by relevant linguistic terms (Figure 8.3). However, the marking mechanism remains unchanged (because, original interface also assumes continuous spectrum for marking purpose).
- Durations of the pre and post-tests are reduced to 30 minutes.
- Questionnaire refined

8.5 Evaluation Process

A group of ten students, seven third year students and three post graduate students, volunteered to participate in this study. Incidentally, all the students were males. They had already completed a basic software engineering course (some of them had completed an advanced course also). The basic course introduces object oriented concepts and the UML notation, whilst the advanced course includes a project requirement which demands high level practical knowledge in using UML for software development processes. They have also studied formal specification using Z language in the advanced course. They had also completed another advanced course which includes material on Object-Z.

8.5.1 Empirical Study

The same processes as in the pilot study were followed in the main experiment. Formal approval for the study was obtained from the ethics committee, and the students completed the necessary formalities. After that, the evaluation process was explained to them, and they were given enough time to clarify any issue regarding the process.

Two different set of MC tests (version 1 and version 2), each containing ten questions of similar complexity, were used in the process (Appendix D). Firstly, a pre-test was administered, and then the students were requested to learn the Object-Z concept 'Visibility List' using the prototype for an hour. After that, a post-test was conducted. As discussed before, the internal validity of the experiment may be affected due to variation in the complexity of the pre and post-tests. To minimize any adverse effects, first of all, the students are randomly allocated into two sub groups (group 1 and group 2) of the same size. For the pre-test, the version 1 of the test was given to the group 1 and version 2 was given to the sub group 2 and for the post-test, the versions were reversed for each sub group. The marks of all the students on both pre and post-tests are given in Appendix D.

The above empirical study is sufficient for evaluating the accuracy of the Learner model. Since Learner model plays a key role in the system, the accuracy of the Learner model influences the efficiency of the whole system. However, as discussed in 8.3.2 the

accuracy of the Learner model alone can be evaluated independently. The next section describes this process in detail.

8.5.2 Log Files for Learner Model Accuracy

Learner model accuracy may be tested at two levels. At the external level, the final estimate of the Learner model and the student's actual post-test scores will be compared. The value of GLA (Global Learning Ability) is the best candidate in this case for the system's final estimate of the learner's ability. However, since the prototype covered only one topic, GLA is updated only once; and therefore, it heavily depends on the learner's own initial estimates about their knowledge levels in the pre-requisite subjects. Therefore, for each student, the average SMS values of the last two mental states of all the sub-concepts are used as the measure of system's estimate of their learning ability. Since the post-test includes MC tests on most of the mental states, the above measure is a reasonably good candidate for the estimate of the Learner model. The post-test scores for each student will be compared against the above measure to evaluate the efficiency of the Learner model.

As mentioned in Chapter 5, this research does not aim for a very accurate Learner model. However, for completeness, the accuracy of the Learner model is tested more minutely, at the internal level, by monitoring the Learner model continuously. Log files are created to store all the relevant information. Two prediction units (with 3 variables and 2 variables) are included to predict the performance of a learner before a formative assessment. The two-variable unit only predicts that a student will perform well or not well; or in other words, especially for MC tests, it would be 'correct' or incorrect'. In addition to these, the three-variable unit may also predict that a student may perform at 'medium' level. The 'medium' level performance makes sense only if CBM scheme is used for MC tests. In terms of accuracy, the three-variable unit predicts better than the two-variable one as the 'medium' level performance can match with both right and wrong answers. The numbers of correct and incorrect predictions from both levels are extracted from the log files (Appendix D). The Learner model's prediction for the learner's performance before each MC test and the learner's actual performance were compared.

8.6 Results and Discussion

This section describes the results of the various evaluation processes, and includes a detailed discussion on the results related to the two key hypotheses and other claims.

8.6.1 Hypothesis– 1: Effectiveness of LOZ

The raw marks of the pre and post-test scores for each student participating in the empirical study are given in Appendix D. The bar chart in Figure 8.4 shows these scores visually. It can be easily seen that the post-test scores are greater than the pre-test scores for all the students. This simply implies that the students have improved after using the prototype for learning. That is, the system is effective for learning Object-Z.

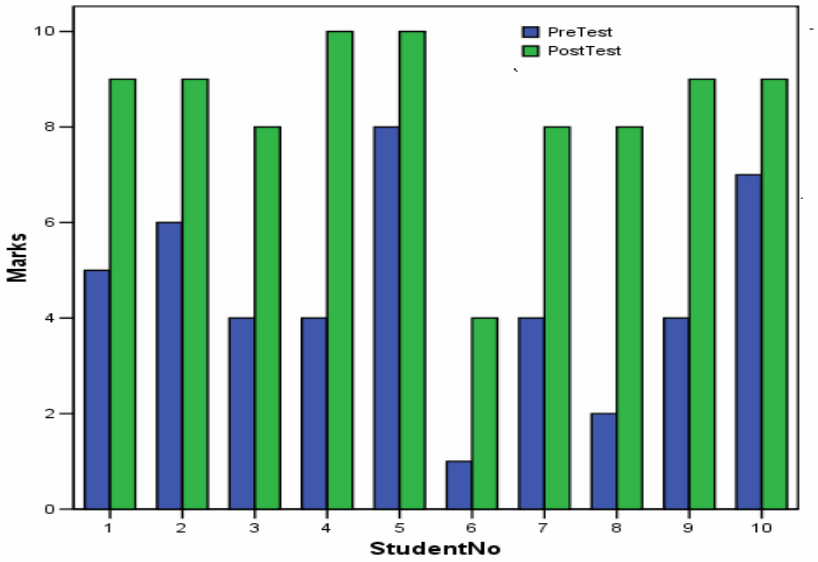


Figure 8.4 Pre Vs Post-test scores- Bar Chart

However, to test whether the variation is statistically significant, a Student-t test for dependent samples is used. Before applying the t-test, it is beneficial to examine the mean and variance of the two samples (see Figure 8.5). The mean of the post-test sample is nearly double that of the pre-test mean, while the variance does not differ much. This observation also supports that the system is efficient for learning Object-Z.

For pair-wise dependent samples, the standard error of the difference between means depends on the correlation coefficient. For high positive correlation the standard error will

be minimum; and therefore, the sample results will reflect the population results in high order (that is, the power of the experiment will be high). Note that the correlation between pair wise scores of both pre and post-tests has significantly high correlation. Therefore the power of this experiment is very high.

Paired Samples Statistics

	Mean	N	Std. Deviation	Std. Error Mean
PreTest	4.50	10	2.121	.671
PostTest	8.40	10	1.713	.542

Paired Samples Correlations

	N	Correlation	Sig.
PreTest & PostTest	10	.734	.016

Figure 8.5 Pre Vs Post-test scores- Basic Statistics

As before, let μ1 and μ2 be the means of the pre and post-test scores of the group respectively.

The related null hypothesis H0: $\mu_1 = \mu_2$

The alternative hypothesis H1: $\mu_1 < \mu_2$

Table 8.3 gives the statistics of differences of pair-wise pre and post scores, and the relevant Student-t estimates.

t-Test: Paired Two Sample for Means	
	Post-test – Pre-test
Mean	3.90
Std Deviation	1.449
Std Error Mean	0.448
df	9
Alpha	0.05
t Stat	8.510
P(T<=t) one-tail	6.729 E-06
t Critical one-tail	1.833

Table 8.3 Difference (post-test – Pre-test) Statistics

At 95% confidence level ($\alpha = 0.05$), the t-value for one tail at the degree of freedom nine is 1.833 (since the alternate hypothesis is one directional, t critical one tail is used), and the estimated t-statistics is 8.51. Since the estimated t value is much greater than the table t value, the null hypothesis can be rejected safely. That is, the alternate hypothesis can be accepted with 95% confidence. In other words, the test ascertains that learning Object-Z using LOZ is effective.

Meta Analysis - Effect Size and Power

The test has established that the learning Object-Z using LOZ gives a statistically significant result. Next, it is useful to examine whether this effect is practically significant. In other words, the magnitude of this effect needs to be estimated. The term Effect Size (ES) is used to refer a collection of indices that measures the magnitude of the effect of treatment in different experiment designs (Cohen 1988). A simple way of determining the effect size of an experiment is finding the standardized difference between the means (called Cohen's d). In this context, it measures how strong is the difference between the means of the test scores before and after learning using LOZ.

Dunlop *et al* (1996) show that the traditional equation that uses t-statistics to calculate the effect size in two independent sample problem ($d = t \sqrt{.}(2/n)$ gives an over estimated value when it is used for correlated samples. They also suggested a formulae to calculate d from t ($d = t \sqrt{.}(2-2r)/n$) - they assumed the variances are same for both samples). The original standard deviations of the pre and post-tests are used below to calculate the effect size

$$d = \frac{(\text{Post-TestMean- Pre-Test Mean})}{\text{SDspooled}}$$

$$= \frac{3.9}{\sqrt{.}((2.93 + 4.5)/2)}$$

$$= 2.023$$

In general, an effect size of 0.8 and above is considered high (Cohen 1988). The effect size of 2 is reported in a classical research by Bloom (1984). Some ITS researchers report an effect size between 0.5 and 1; for example, Albacete *et al* (2000) reports 0.63, Anderson *et al* (1995) reports 0.1, Mitrovic *et al* (2002) reports 0.66, and Suraweera *et al* (2004) reports 0.63.

The power measure, the probability of rejecting the null hypothesis while it is false, is used while designing the experiment to determine the effective sample size for an anticipated effect size. Retrospectively, the power measure may be used in meta analysis to measure the effect of the results of an experiment. For the sample size ten and the effect size 2.023, the power of the experiment is 0.99 at significance 0.05 (Figure 8.6). In other words, if the experiment is repeated under the same conditions, the same significance will be obtained almost all the time. Chin (2001) recommends that the power should be 0. 8. Suraweera *et al* (2004) reports 0.75 (at significance 0.05) for their CBL system.

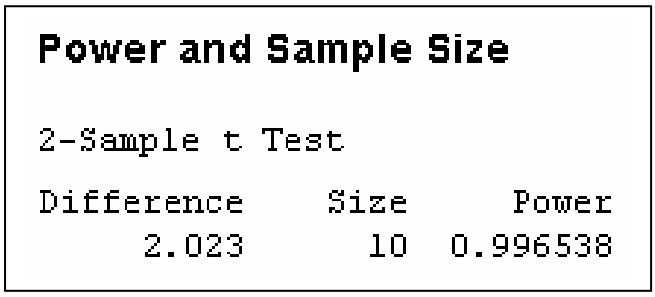


Figure 8.6 Power of the Experiment

Extraneous Variable Control

As mentioned before, the test questions in version 1 and version 2 are carefully selected so that they are of similar complexity. Moreover, in order to avoid any variation in the complexity level of pre-test and post-test, both sets version 1 and version 2 were given in both tests. This arrangement ensures the internal validity of this experiment. A test can be carried out to see whether there is any statistically significant variation in the complexity level of the pre and post-tests. For each subject, a pair of pre and post-test scores is available. First, the mean and standard deviations of the samples may be examined (given in Figure 8.7).

It can be seen that, the means of both version 1 and 2 are nearly equal. This indicates that the difficulty levels of both versions of the tests are almost same. The variances are high for both versions because half of the scores of both versions are obtained in pre-test and other half in post-test. The mean of the pair wise difference (ver1 – ver2) is nearly zero. The calculated t-statistics (-0.073) is lower than the table t-statistics for non-directional case (0.044). Therefore, the null hypothesis is retained, That is, there is no statistically

significance difference between the complexity of version 1 and version 2 of the tests. Note that, the correlation between pre-test and post-test scores for each participant is very low. However, this shows only insignificant linear correlation. It is not strange as both versions are given alternatively in both pre and post-tests. Actually, there can be a strong non linear correlation.

Paired Samples Statistics

	Mean	N	Std. Deviation	Std. Error Mean
Ver1	6.40	10	3.373	1.067
Ver2	6.50	10	2.121	.671

Paired Samples Correlations

	N	Correlation	Sig.
Ver1 & Ver2	10	-.217	.546

Paired Samples Test Ver1 - Ver2

Paired Differences					
Mean	Std. Deviation	Std. Error Mean	t	df	Sig. (2-tailed)
-.100	4.358	1.378	-.073	9	.944

Figure 8.7 Statistics for Tests - Ver. 1 Vs Ver.2

As mentioned before, the subjective evaluation process complements the empirical evaluation. In the context of hypothesis 1, there are many items in the questionnaire that directly or indirectly relate to the overall effectiveness of the system. The results of the subjective evaluation and the related discussion are given in Section 8.6.3.

8.6.2 Hypothesis – 2: Accuracy of the Learner Model

As mentioned before, there are two types of tests conducted to evaluate the second hypothesis. The first test uses the final estimate of the Learner model, after the learning process is completed. The post-test scores of the students are matched against the final Learner model estimate. The Learner model's estimate of a learner's learning ability is given by the average SMS values of the last two mental states of all the sub-concepts. The

second test uses a series of estimates made by the Learner model during the learning process. For each student, the Learner model's estimates for his performance on each MC tests are compared with his actual performance.

In the first test, the Pearson correlation between the post-test scores of each student and the system's estimate for their learning ability gives a measure for the degree of linear relationship between those two variables; and this measure, in turn, gives a measure for the degree of accuracy of the Learner model.

For a two-tailed test of the correlation:

Let, $\mathbf{H_0}$: $\rho = 0$

$\mathbf{H_1}$: $\rho \neq 0$, where ρ is the correlation between a pair of variables.

The Pearson product moment correlation coefficient between post-test scores and average SMS values were calculated and found to be very high ($r = 0.827$ – see Figure 8.8). There is a high correlation between the system's estimate about the learner's knowledge level and their actual knowledge level (if post-test scores are considered as evidence).

.

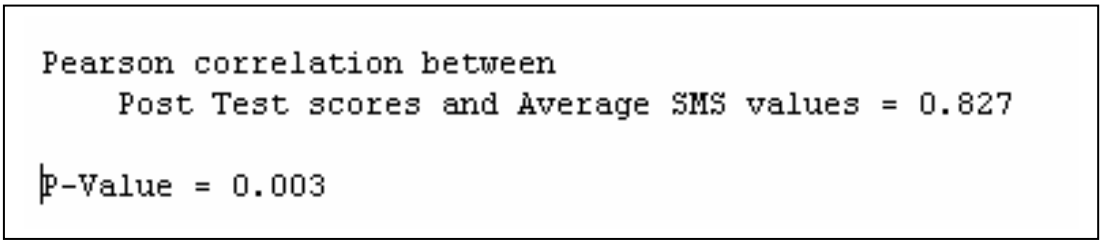


Figure 8.8 Pearson Correlation Co-efficient

In the second type of test, the system's predictions for each student for each MC test were extracted from the log files. There were two levels of prediction units used for testing. The two-level unit predicts whether the learner will succeed or fail the next MC test. In addition to the above two values, the three-level unit includes also a 'middle' option. The system's estimates and the learner's actual performance for each MC test were compared. The numbers of correct and incorrect predictions for each student were recorded. On average, the 3-level unit predicts nearly 84% of the cases successfully (with SD =11.5), and the 2-level unit correctly predicts nearly 55% of the time only (Table 8.4).

Performance Vs Prediction		
Measure	**% correct prediction**	
	3-Level	***2-Level***
Mean	84.167	55.174
Standard Deviation	11.535	12.744
Range	30.780	46.429

Table 8.4 Learner Model Accuracy - Statistics

The clustered bar charts show the percentages of correct predictions against incorrect predictions for both 2-level and 3-level units (Figure 8.9 & 8.10). The bar charts clearly show that the 3-level prediction unit successfully predicted the performance most of the times for all the students; in contrast, the 2-level prediction failed to predict the performance sufficiently. The above result is expected. The entire Learner model is based on the fuzzy mechanism, and therefore, it is difficult for such a fuzzy unit to predict one of the two values precisely.

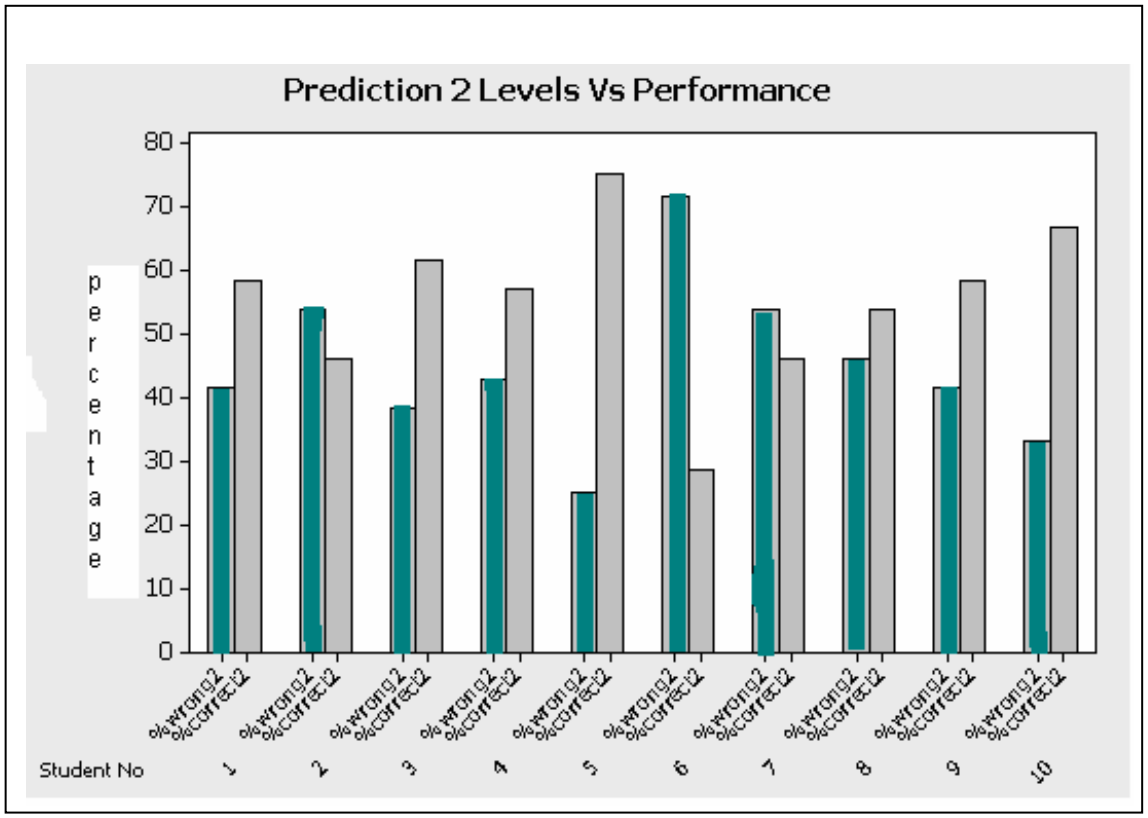


Figure 8.9 Performance Prediction Accuracy: 2-level

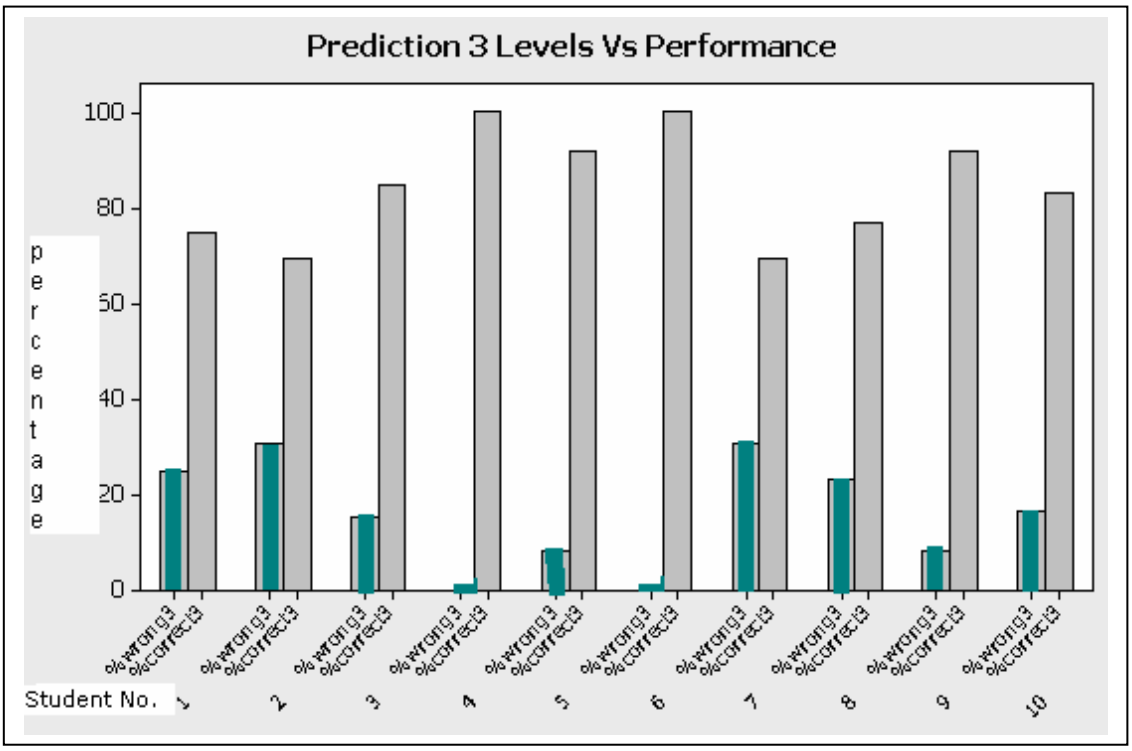


Figure 8.10 Performance Prediction Accuracy: 3-level

8.6.3 Evaluating CBM Schema

As discussed earlier, the traditional MC test is not sufficient for assessment and therefore a confidence based MC test schema (an interface and a marking scheme) is proposed in this research. In order to evaluate the proposed CBM schema, two interfaces - traditional and CBM based (Figure 8.3), were provided with each test item. The original plan to compare both methods statistically was later withdrawn. Nevertheless, the results of subjective evaluation for this claim will be discussed later in this chapter. The reason for the withdrawal is discussed below.

Based on the answering pattern, it can be considered that the interface of CBM schema confused some students. Two issues may be contributed to this situation. First of all, the linguistic terms used for different scales do not show their numerical levels explicitly (except their order of appearance). For example, some students did not distinguish between the terms ‘impossible’ and ‘not possible’ (or considered ‘impossible’ is stronger denial than ‘not possible’ – regardless of their order), and therefore they had selected ‘impossible’ for all other options when they were ‘certain’ that a particular option is correct. Natural language is inherently ambiguous, and therefore it can be concluded that

using linguistic terms in place of numerical scales for this type of interfaces could cause adverse effects.

Secondly, the answer options did not go well with the interface. This situation might have been avoided if the marking mechanism had been made explicit to the students well before the tests. For some MC test items there were more than one correct option, and still for some other test items none of the options were correct. The answer options are included to indicate these cases; for example, all the answers are correct or none of the answer is correct or combinations such as (a) and (b) are correct. The traditional MC tests usually have this type of questions, but students will not be confused as they need to select only the best answer. In contrast, for CBM test, since all the options can be marked with different confidence level, the students get confused.

As a result, the statistical analysis based on this evaluation process may not be accurate, and hence, it will not be reported further in this research. However, the subjective evaluation on CBM schema was carried out and the results will be discussed in the next section.

8.6.4 Subjective Analysis

A questionnaire was administered to collect quantitative and qualitative data about the student's opinions about the prototype. The responses of the participants are tabulated in Table 8.5. The percentages given in the following paragraphs refer to the percentages of answers in the bottom or top two points of the five point scale as relevant. The middle point is considered as a neutral response.

Effectiveness of LOZ

Items 1 to 5, and then 12 and 13 in the questionnaire (Table 8.5) relate to the effectiveness of LOZ (hypotheses 1) and hence it complements the empirical evaluation process. In particular, items 12 and 13 directly measure the acceptance and satisfaction levels of the participants. The following observations indicate that the system is effective for learning Object-Z notation.

- For item 12 in the questionnaire (Table 8.5), 90% of the students (with Mean = 4.2 and SD = 0.63) agreed that the system provides a suitable environment to learn Object-Z concepts (one student remained neutral).
- For item 13 in the questionnaire (Table 8.5), 90% of the students (with Mean = 4.0 and SD = 1.15) indicated that they are satisfied with the system, and agreed that they would recommend the system to their friends to learn Object-Z (however, one student strongly disagreed).

There are some open-ended responses of interest. Many participants noted that the GUI and the response time need to be improved. In fact, the delay in response was not due to the calculations performed in the Learner model, but due to the inefficient data base access. This condition often occurs in a prototype.
Two other encouraging comments are;

- *The functionality is good. It's a very useful piece of software.*
- *Found this to be a really easy way to learn Object-Z*

Accuracy of the Learner Model

Item 4 in the questionnaire (Table 8.5), which refers to the feedback of the system, was included to test the accuracy or effectiveness of the Learner model (hypotheses 2). The system is supposed to provide suitable feedback to the individual students based on its Learner model. In other words, if the Learner model is accurate, the system should provide suitable feedback tailored to the individual learners.

The following observation indicates that the system provides suitable feedback, and therefore the Learner model should be reasonably accurate.

- 90% of the students (with Mean = 4.3 and SD = 0.67) considered that the format and content of the feedback as appropriate (Table 8.5).

Moreover, the following comments made by the participants with reference to the Learner model are encouraging.

- *The user feedback screen after successfully answering a question contains good, useful information,*
- *Like the careless mistake, get given the question again.*

- *Having the intelligent feedback system is useful; it doesn't give the answers away unless it is obvious that you don't know the answer.*

No	Group	Related Issue	Statement (A short description)	Disagree %	Agree %	Mean	SD
1	FoPSI	Pre-conditioning Phase	Pre-conditioning phase is helpful	0	100	4.3	0.48
2	FoPSI	Scaffolding Strategy	Scaffolding is handled effectively	0	90	4.3	0.67
3	FoPSI	Exploration Phase	Exploration feature is useful	0	40	3.5	0.71
4	FoPSI	Feedback in Learner Model	System provides adaptable feedback	0	90	4.3	0.67
5	FoPSI	Refinement in Learner Model	Providing Java code motivates learning	10	50	3.4	0.70
6	Opening LM	Opening Learner Model	Viewing System's estimate useful	10	70	3.8	0.92
7	Opening LM	Modifying Learner Model	Facility for modifying the estimate is handy	10	40	3.3	0.67
8	Opening LM	Modifying Learner Model	Feature to change the feedback level is good	0	40	3	0.94
9	Opening LM	Modifying Mentor Model	Facility to modify scaffolding steps is useful	0	80	3.8	0.42
10	CBM	MC test strategy	CMB allows to express partial knowledge	0	100	4.8	0.42
11	CBM	MC test strategy	CBM is better compared to traditional scheme	10	90	4.3	0.95
12	General	Learner Acceptance	System gives suitable learning environment	10	90	4.2	0.63
13	General	Learner Satisfaction	I will not hesitate to recommend it to others	10	90	4	1.15

Table 8.5 Subjective Evaluation – Response Statistics

Effectiveness of the FoPSI Model

Items 1, 2, 3 and 5 in the questionnaire (Table 8.5) are included to obtain quantitative data on the third claim (the FoPSI model is effective for instruction).

The following observations are made based on the responses on the other four items:

- Item 1: All the students agreed that the preconditioning stage in the FoPSI model is useful for learning (with Mean = 4.3 and SD = 0.48).
- Item 2: 90% of the students agreed that their learning ability is enhanced by scaffolding (with Mean = 4.3 and SD = 0.67).
- Item 3: While 60% of the students remained neutral, only 40% agreed that the exploration phase is useful for learning (with Mean = 3.5 and SD = 0.71).

- Item 5: Only 50% of the students found that that the Refinement unit is useful (with Mean = 3.4 and SD = 0.70). Two students did mention that they had not even noticed this facility.

Effectiveness of Opening Learner Model

The fourth claim is related to opening the Learner model to students. The second section of the questionnaire (Table 8.5) includes four items (statements 6-9) to perform subjective evaluation on this claim. The current version of the prototype only includes a facility to browse the contents of the Learner model. Learners may inspect the system's estimate of their knowledge level.

The following observations are made based on the responses on items six to nine:

- Item 6: 70% of the students agreed that viewing their current estimate is very useful for learning (with Mean = 3.8, with SD = 0.92).
- Items 7 and 8: Only 40% of the students thought that modifying learner model is useful.
- Item 9: 80% of the students speculate that modifying mentor model would be useful (with Mean = 3.8 and SD = 0.42).

Effectiveness of CBM Schema

Items 10 and 11 are included in the questionnaire in order to perform the subjective evaluation on the proposed CBM schema (Table 8.5). The responses are encouraging.

- All the students agreed that the proposed CBM schema allowed them to express their partial knowledge efficiently (among them 80% of them strongly agreed).
- When comparing the proposed schema to the traditional MC tests, 90% of the students prefer the former to the later.

8.6.5 Summary of Results & Discussion

The findings based on the empirical and subjective evaluation analysis are summarized in the following paragraphs.

Hypotheses 1: LOZ is effective for learning Object-Z

According to the empirical study the effectiveness of the prototype for learning object-Z is statistically significant. Based on the Student-t estimate for dependant samples, the t-value at 95% confidence level ($\alpha = 0.05$) for one tail at the degree of freedom nine is 1.833 and the estimated t-statistics is 8.51. The meta-analysis revealed that the experiment has a high effect size 2 (recommended value is 0.8) and a high power 0.9 (recommended value is 0.8). The effect of extraneous variables on the experiment also found to be minimal. In addition to the empirical study, the subjective analyses also confirm the system is effective to learn Object-Z. A high percentage (90%) of the participants mentioned that the system provides a suitable environment for learning Object-Z. Finally, participants also include encouraging comments on the accuracy of the Learner model and on the overall effectiveness of the prototypes. Finally, participants also include encouraging comments on the overall effectiveness of the prototype for learning.

Hypotheses 2 : The Learner Model in LOZ accurately model the learner

The empirical study confirms that the Learner model's accuracy on modelling the learner's knowledge level is statistically significant. There is a high correlation between the system's estimate about the learner's knowledge level and their actual knowledge level. The evaluation is conducted at two levels. At a coarse level, the post-test scores after the learning process was completed are considered as evidence for learner's knowledge level. The Pearson product moment correlation coefficient between post-test scores and average SMS values are calculated and found to be very high ($r = 0.827$). At the finer level, during the learning process, the learner's performance on each MC tests is considered evidence for his/her knowledge level. The system also records its predictions before each test. The numbers of correct and incorrect predictions for each student were analysed. On average, the system predicted nearly 84% of the cases successfully (with SD =11.5). The effect of extraneous variables on the experiment was also found to be minimal. The subjective analyses confirmed the findings of the empirical study. A high percentage (90%) of the participants mentioned that the system provides suitable feedback. Finally, participants also include positive comments on the accuracy of the Learner model.

Other Claims:

The subjective analysis is the only means of evaluation conducted for the last three claims. Regarding the FoPSI model, all the participants mentioned that the pre-conditioning stage were useful for learning Object-Z, and a high percentage (90%) mentioned scaffolding process were useful. Regarding opening the Learner model, 70% of the participants found that the feature for viewing the Learner model is useful, and 80% supported the idea of opening Mentor model. Finally, regarding the CBM, again a high percentage (90%) of participant preferred CBM schema to the traditional schema, and 80% believe that the CBM schema allows them to express partial knowledge on a test item.

The exploration phase in FoPSI model was not fully implemented in the prototype. Therefore it is not strange that 60% of the participants marked neutral for item 3 of the questionnaire (Table 8.5). Further, the Refinement unit is not useful in MC test scenario, and therefore, the Refinement unit in the prototype includes simple one-to-one mappings from Java code to Object-Z schemas. Merely 50% of the students found this feature useful. Regarding modifying Learner model, the prototype only include user interface with limited functionality. Again 60% of the students remained neutral on this issue (item 8 on Table 8.5).

Sample Size and Accuracy

A sample of size ten (repeated samples and pair-wise t-test) is used in this evaluation. For small samples, there is the risk of accepting the null hypothesis while the alternate hypothesis is true. On the other hand, large samples are required for high accuracy. Alpha cannot be compromised for small sample sizes. However, for small standard deviations, a small sample size is sufficient for reliable testing. If the pair-wise t-test is used with a small sample size, a high mean difference (e.g. nearly two times the standard deviation for power 80%) is required to reject the null hypothesis safely. The required sample size for an expected set of values (means, standard deviations, power and alpha) can be calculated. For the mean difference 3.9 and standard deviation 1.93 with alpha 0.05 and power 0.99, the required size of the sample for a two-tailed t-test is calculated as 8 (the calculator used for this is at http://www.stat.ubc.ca /~rollin/stats/ ssize/n2.html - accessed

on May 29, 2008). It can be noted that, with the sample size of ten, the same values are obtained for means and standard deviations at the same significance.

Advantages of True-experiment Design

Pre-experiment design is used in this evaluation. True-experiment design is more flexible (see Section 8.3.1). In pre-experiment design one can argue that whilst a change in the outcome or dependent variable has taken place, the change might have occurred even without the application of the treatment. For example, in this study, it is possible that merely thinking about the pre-test questions could cause the change in the grades in the post-test. This issue is not of concern in a true-experiment as the control group has also taken the pre-test. It is also possible to give different alternate treatments (for example, text books only, lectures, or the same software with the learner model disabled) to the control group, and the effects may be compared with the experimental treatment.

8.7 Summary

The prototype described in Chapter 7 was used to evaluate the key research proposals made in this research. In particular, two hypotheses are considered important and related. The first hypothesis is about the effectiveness of the overall strategy proposed for designing CBL systems for complex domains using problem transformation. For practical reasons a pre-experiment design is used for objective evaluation. The second hypothesis is about the accuracy of the Learner model. Both are evaluated using empirical study and subjective analysis.

A group of ten students were given a pre-test, and then asked to learn a concept using the prototype. Later, a post-test was given. The performance of the students on both tests was statistically analysed. The Student-t statistics for paired-samples was used at 0.05 significant level. The test reveals that the post-test scores are significantly higher than the pre-test scores. Another test confirmed that both the pre and post-tests were of the same level of difficulty. The meta-analysis validated the test results. The effect size of the test was greater than 2 and the power is greater than 8. Both values are reasonable compared to the similar research reported in the literature.

In order to evaluate the second hypothesis, two tests were performed. The first test examined whether the final estimate of the Learner model for a student matches with his/her post-test score. The Pearson test asserted that the correlation between both scores is high (Pearson r = 0.85). In addition to that, another test was designed to test whether the system continuously models the student sufficiently accurately. For this, relevant changes were made to the original design in order to predict the expected performance for a student on each MC test. There were two types of predictions made; two-levels and three-level. The two-level only predict whether a student will answer correctly or incorrectly. The three-level also includes 'medium' option. After each time a student answers an MC test, the system writes log files to store the expected performance and actual performance. The analysis shows that the three-level prediction mechanism predicts the performance accurately for more than 80% of the tests.

In addition to the empirical test, an evaluation of opinions was also conducted by administering a questionnaire. Some items in the questionnaire refer to the first two hypotheses. For other claims, the questionnaire is the only means of evaluation reported in this research. In support of the first hypothesis, ninety percent of the students agreed that the system gives suitable environment to learn Object-Z concepts, and they would recommend LOZ to their friends. Related to the second hypothesis, ninety percent of the students considered the feedback as appropriate. All the participants confirmed that preconditioning stage in the FoPSI model is useful for learning. Ninety percent mentioned that the scaffolding strategy enhance their learning ability. All the students agreed that the proposed CBM schema allowed them to express their partial knowledge efficiently. Ninety percent of the participants prefer the proposed CBM schema compared to the traditional MC tests. Finally, participants also include encouraging comments on the accuracy of the Learner model and on the overall effectiveness of the prototypes.

Chapter 9

CBL system for Object-Z – Phase II

9.1 Introduction

In the previous three chapters, issues relating to the first phase of the design of a CBL system for Object-Z notation are discussed. The first phase covers the fundamental features such as the syntax and semantic of the notation. Having learned the fundamentals, students need to practice applying their knowledge to real problems. They should be able to create formal requirement specification documents using Object-Z notation. The second phase deals with the issue of designing an adaptable environment for the learner to practice the process of creating Object-Z specifications. In this chapter, the issues relating to the second phase are discussed. This chapter is included for completeness- a full design is out of scope of this research. The key design artefacts proposed for phase I may be reused in phase II with suitable modifications. Section 9.2 describes the proposed enhancements and modifications.

Phase two of the design uses problem transformation as a way to provide a constructive learning environment. Scaffolding is used to support students in two ways: firstly, while performing abstraction (relevant UML models are used as scaffolds), and later, while constructing complex structures in Object-Z notation. How to transform UML models to Object-Z schemas is the key question in phase two. In particular, transforming operation specifications to relevant Object-Z schemas is discussed in detail. An iterative use case based step-by-step methodology is proposed in this research for this transformation process. This process is systematically used to design the active learning environment in phase two. Section 9.3 illustrates the proposed transformation methodology using a simple case study.

9.2 A CBL System for Object-Z - Phase-II

The proposed CBL system for phase-II is primarily a practice environment. There are many practice environments reported in the literature with varying degree of adaptability and learner control. For example, LISPITS (Anderson *et al.* 1985) provides interactive help based on model tracing: A later version improves the Learner model based on Dynamic BN theory (Corbett *et al.* 1992). BRIDGE (Bonar *et al.* 1986) helps learners to view the problem in a declarative manner and leads them to a procedural solution. PROUST (Johnson *et al.* 1985) tries to infer the intentions from the students' completed solutions. The domain model of PROUST contains a rich set of models for both correct and incorrect intentions derived from novices' solutions for real programming exercises.

The architecture of the system is similar to the earlier design: however, the functionality and the contents of the constituent models are considerably enhanced. The following sections describe the proposed design for the key constituents in the second phase. Being a practice environment, case studies and examples play important roles in this phase. The case study given in Chapter 4 is used here for illustration.

9.2.1 Mentor Model

The Mentor model includes constructs that support teaching strategies. In phase II, scaffolding is used in two ways. Firstly, three scaffolding levels are used to help the students deal with abstraction problem. In addition to abstraction, creating complex Object-Z structures is challenging. In order to help the students to construct complex Object-Z structures incrementally, there are two or more scaffolding stages included at each level.

During the first scaffolding level, the relevant UML models for structural and functional abstractions of the case studies will be provided along with the textual descriptions. At least the use case model, sequence diagram, class diagrams, and operation specifications will be provided. In the mean time, in order to help the students create complex Object-Z constructs, different levels of structural support will be given. During the second level, only the structural abstraction will be given. The sequence diagram and operation

specifications may be provided on request. During the third level, only the textual descriptions will be given. However, the whole UML specifications may be accessed by a student at will. Similar to phase-I, this arrangement helps students with abstraction, and therefore allows them to concentrate on mathematical aspects of the notation. But unlike phase-I, all the assessments are of critical response type (not MC tests) and require them to create their own solutions.

Constructing a solution is a higher order mental activity than selecting the correct solution. The required degree of creative activity involved in the problems may vary from specifying a single predicate to creating an entire Object-Z schema. The proposed scaffolding strategy will be illustrated in the following paragraphs using the case study (Figure 9.1).

Case Study

A building has many rooms. Each room has at least one key (a magnetized card).A key can open only one room. At any time, a room may be in a locked or unlocked state. Opening a given room is considered the main requirement of this system.

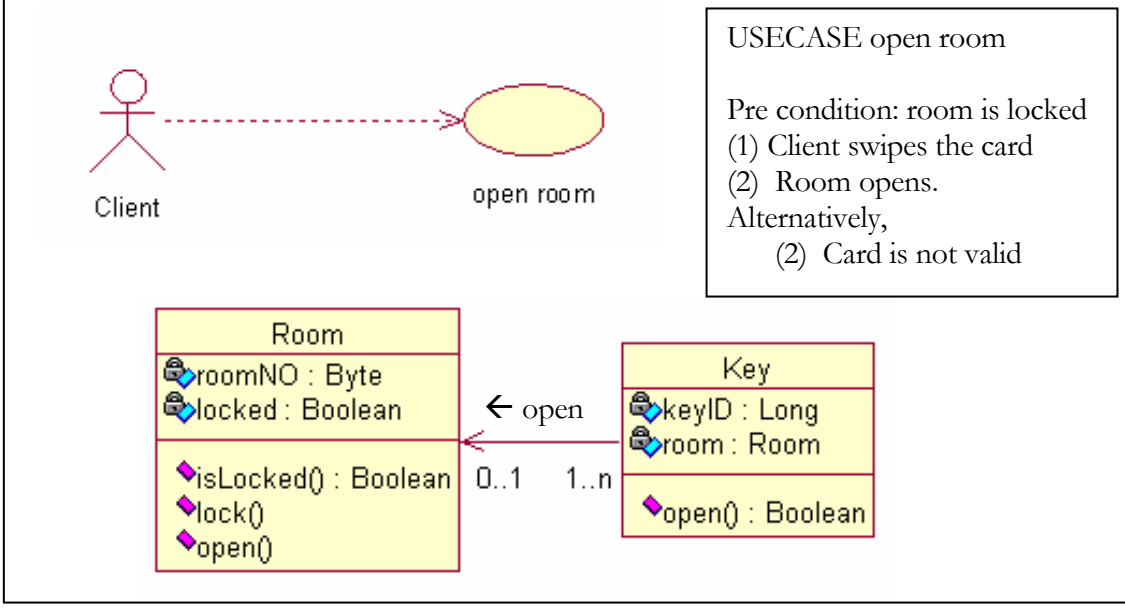


Figure 9.1 Case Study

Scaffolding Level –1: Supports Data and Functional Abstraction

At the initial stages of the first level, the system will provide help not only on data and functional abstractions but also on the syntactic and structural aspects of the notation. For example, at the first stage of the first level, the learner may be asked to create Object-Z specifications related to the use case 'open room'. At this level, all the UML structures such as the use case model (Figure 9.1), sequence diagram (Figure 4.4), class diagrams (Figure 9.1), and operation specifications (Figure 4.3) will be made available to a learner. The system may use the proposed step-by-step algorithm (described later). Initially, for example, the complete object schema may be provided except for a place holder for a single predicate, which the learner would be required to fill-in (see Figure 9.2). The next scaffolding stage of the first level may require students to create a whole predicate part of a schema.

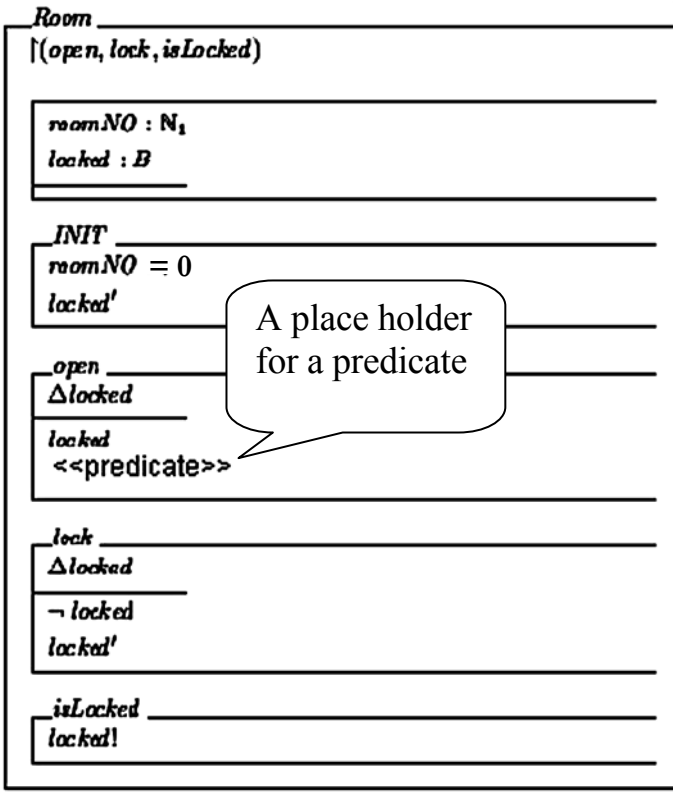


Figure 9.2 Scaffolding Level-1: Support for Syntax

Initially, the learner is required to understand the given structure and devise the required predicate (or predicates) to fill-in. In this case, as the choices for the learners are limited, the model tracing approach is suitable for instruction (Anderson *et al.* 1985).

Nevertheless, unlike programming[7], specification is declarative. While creating specifications, students may not think procedurally, and therefore, may not create the parts of the solution in sequence. Therefore, if students are required to create a specification involving a substantial amount of structure, the symbol by symbol tracing approach is not appropriate for mentoring; instead, a concept tracing approach (Brusilovsky 1995) is proposed at this level. The system should give some leeway so that the learner can complete a part of the solution freely. For example, if a learner is asked to write the whole predicate part, the system should allow the learner to complete at least a single predicate before tracing the solution: it should not start to coach students at their first (perhaps a careless) mistake. For each case study, the possible problem solution paths and potential error paths need to be included in the system. Expert judgments, empirical studies and cognitive task analysis may be used to identify these models.

The system will provide an interface so that the required symbols and potential identifiers may be selected from given lists. Learners have to construct the solution by joining the relevant parts. Free-form typing is avoided in the early stages. This approach has two advantages. For the novices, it allows them to concentrate on a part of a problem that they can handle at a time. There are also computational advantages. Syntax checking is unnecessary as the structure of the schema is already given, and parsing is not required as there is no free-form typing involved. Furthermore, since the learner's freedom is considerably limited and concept tracing approach is used, diagnosing is also not complex (though, it will be more difficult than symbol by symbol tracing).

Gradually, the system's support for structural aspects of the notation will be withdrawn, and eventually, at the later stages of the first level, the system will provide only the state schema and INIT schema automatically. The learner may still be able to access all the UML diagrams. However, they need to create the operation schemas from scratch. They may follow the proposed bottom-up algorithm (described in Section 9.2), or may proceed in their preferred order. Nevertheless, the names of the classes, attributes, and operations have to be selected from the given lists (Figure 9.3).

[7] Even CBL systems for declarative programming languages tend to teach procedural aspects of the languages (for example, LISPITS (Anderson *et al.* 1985)).

The granularity of the concept involved in diagnosing may gradually change from fine to coarse: for example, from a single predicate to an entire schema. At the later stages, similar to debuggers in CBL systems for programming, the whole solution which involves a collection of object schemas may need to be analysed. However, interactive help may be provided during problem solving (Figure 9.4). Nevertheless, since the learner is given more freedom, the system now should include at least a syntax sensitive editor, a parser, and a relatively more complex diagnosing unit than the early stages.

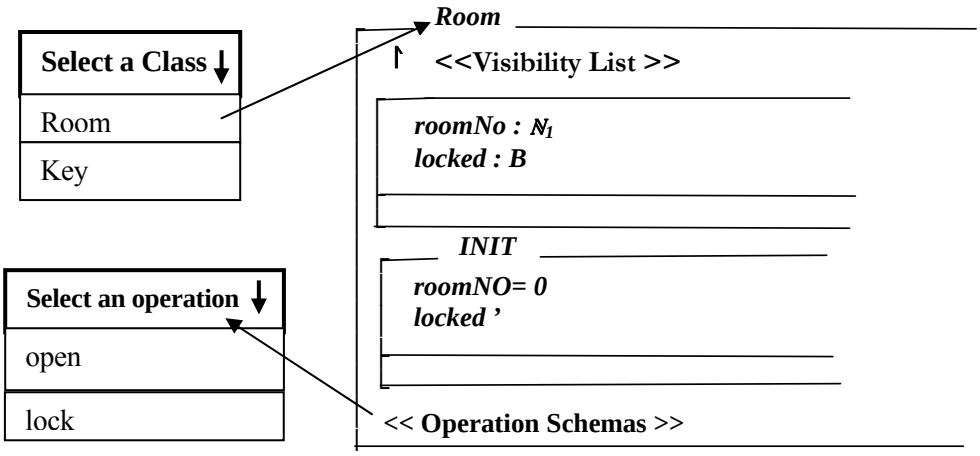


Figure 9.3 Challenge: Specifying Functional Abstraction

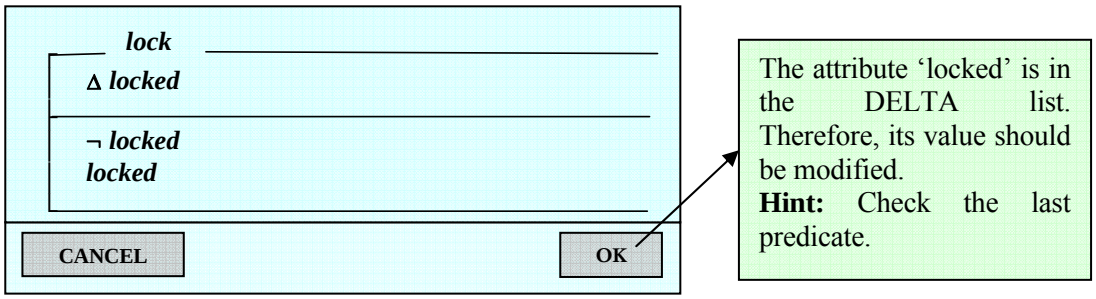


Figure 9.4 Interactive Help on Semantics

Scaffolding Level – 2: Supports Functional Abstraction Only

During the second level of scaffolding, only the data abstraction will be provided. The learner is expected to perform functional abstraction and then to create relevant Object-Z structures. The class diagram will be provided (but, without any method). However, the class invariants, if any, will be provided. The use case model will be available for the

student. Similar to the first level, at the initial stages, the system will provide the structure of the schemas: the state schema and INIT schema will be automatically included. However, unlike the first level, none of the operation schemas will be provided at this level. The key task for learners at this stage is assigning responsibilities between classes. They will be encouraged to sketch suitable sequence diagrams and operation specifications offline. They may check their models with sample solutions of the system.

Once a realization plan is devised for the given use case, students are encouraged to follow the proposed bottom-up algorithm in order to construct the related operation schemas. The names of the classes and attributes may be selected from the given list (see Figure 9.5). Note that, unlike level one (Figure 9.3), the names of the operations are not readily available, but the user should give suitable name for an operation at this level.

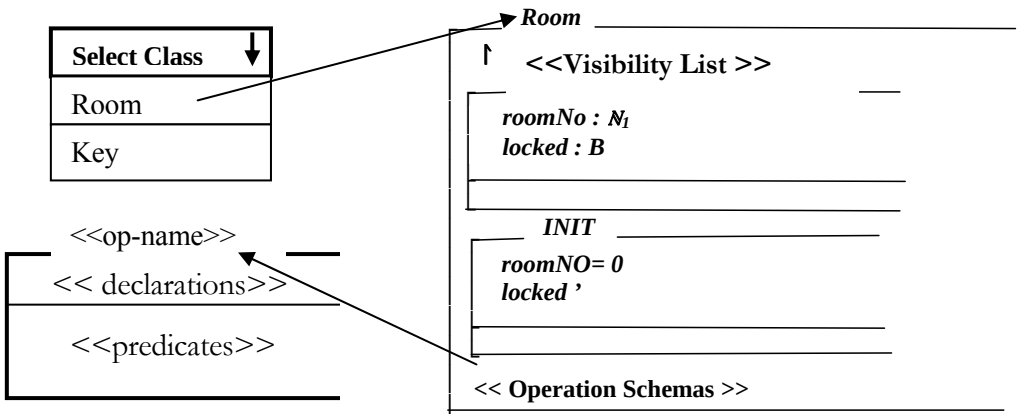


Figure 9.5 Challenge: Functional Abstraction

Gradually, support for the structures will be withdrawn. At the final stages of the second level, except the class name, only the place holders for all the constituent schemas will be automatically included in an object schema (Figure 9.6). Since the class diagram is given, completing the state schema and init schema will not be very difficult.

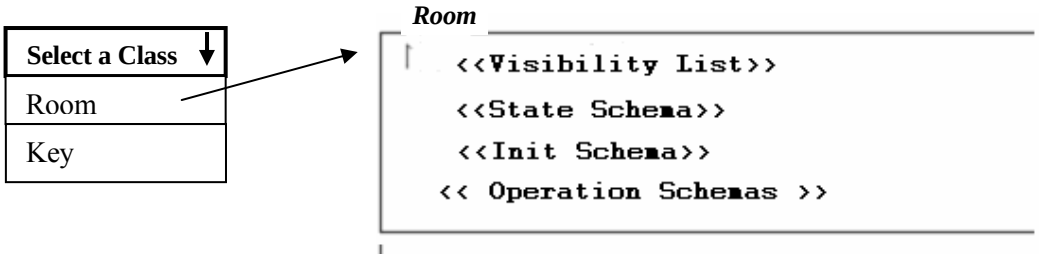


Figure 9.6 Challenge: Data Specification and Functional Abstraction

Scaffolding Level – 3: Supports neither Data nor Functional Abstraction

During the third scaffolding level, there will be no help available on either abstraction. However, similar to previous levels, some help on structure and syntax of the Object-Z notation will be available at the early stages. The class schema may provide place holders for its constituents. These place holders, when selected, would expand further to give their own structure (Figure 9.7).

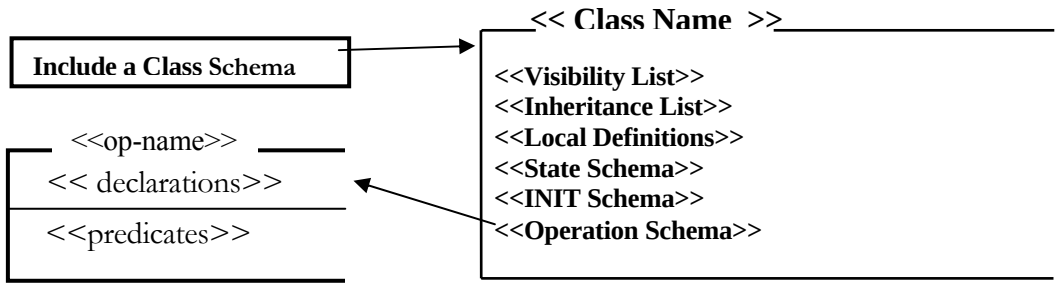


Figure 9.7 Challenge: Data & Functional Abstraction

At the later stages, the system will not give assistance on either abstraction or structure. The learner needs to construct the entire class schema from the beginning. On the other hand, there will be no restrictions. The students may use their own abstraction and may create the related specification in their preferred order. Before creating an Object-Z specification, they are encouraged to create their own UML models such as class diagrams and sequence diagrams offline. The system will also have some sample solutions which can be inspected by the learner at will.

Formal specifications, being declarative in nature, have more flexibility of ordering than procedural programming. Therefore, a graphical editor, which facilitates learners to create specifications in their preferred order, should be included. A syntax sensitive parser is also essential for free-form editing. In order to provide interactive response in reasonable time, the parser should be powerful enough to handle the resultant document incrementally. Since the constituent structures can be added in different orders, providing interactive help on semantic matters will be challenging. However, since mathematical constructs are involved, it may be possible to recognize discrepancies at the early stages of the construction process.

9.2.2 Domain Model

In addition to the domain model designed earlier for phase-I, the domain model in phase-II includes a number of relevant case studies for each main-concept. Case studies are carefully selected to suit the scaffolding stages. Each case study will be associated with a collection of testable concepts (Figure 9.8). For example, in the case study in Figure 9.1, the operation schema 'lock' should contain attribute *locked* in its DELTA list: it is a testable concept. Each testable concept will be associated with one or more sub-concepts.

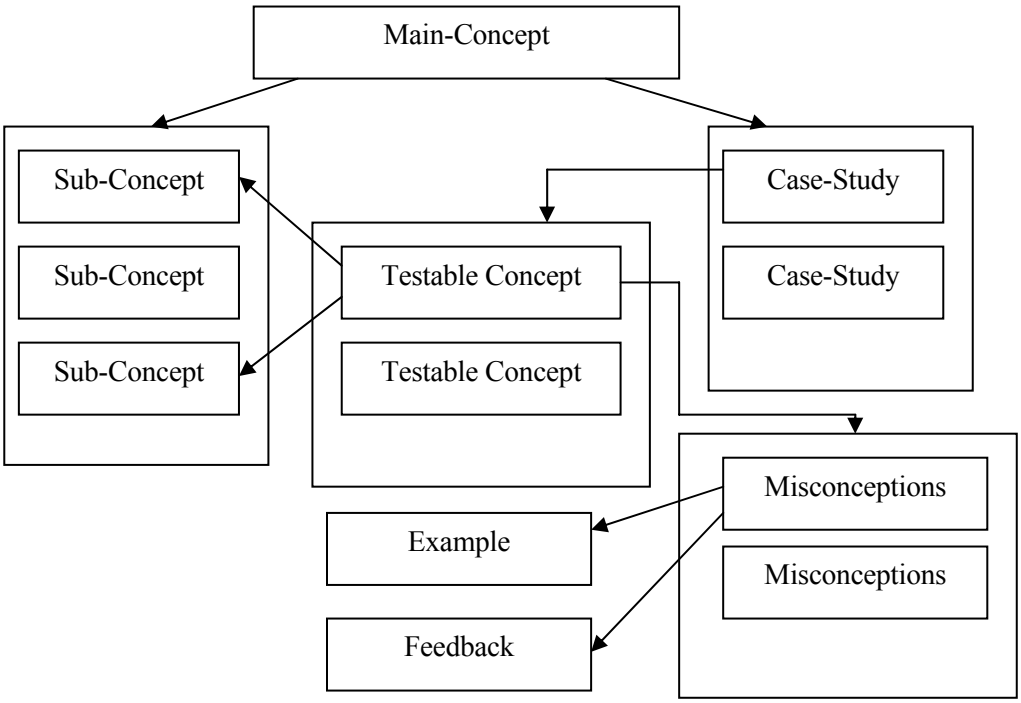


Figure 9.8 A portion of Domain Model for Phase-II

Potential misconceptions will be identified for each testable concept. This will enhance the collection of misconceptions in the previous phase. One or more worked examples and relevant feedback material will be included for each misconception.

9.2.3 Learner model

There are two important types of pedagogical actions available in phase-II: task sequencing and selecting appropriate interactive support during problem solving. Task sequencing, in the context of phase-II, refers to the act of selecting an appropriate case study for a learner.

Task Sequencing

The Learner model proposed in phase-I is used to provide suitable feedback, and then to select the appropriate MC test. It can be slightly modified for task sequencing. Compared to phase-I, the task sequencing in phase-II works at a higher level. Instead of selecting an MC test associated with different scaffolding stages of a sub-concept, the Learner model in phase-II should be able to select a suitable case study relevant to different main-concepts. Therefore, the measure GLA (General Learning Ability) may be used in the dynamic causal network (Figure 9.9), and SMS (Strength of Mental State) will be used for interactive problem solving help. However, a suitable metric to determine the performance of a student (in solving a case study) should be devised. Each case study will be assigned a difficulty level D. The overall performance, P, of a student on a case study may be predicted based on GLA and D. Fuzzy rules may be designed for this relationship (Table 9.1). GLA is dynamic, and will be predicted only after a case study is completed. Similar to phase-I, BN theory may be used for dynamically predicting (Table 6.7) and for updating GLA, but in latter case, the actual performance will be used as evidence.

The actual performance of a student may not alone reflect the exact ability of a learner. Therefore, based on both predicated and actual performance (Figure 9.10), relevant fuzzy rules may be designed to handle uncertainty while determining the suitable next scaffolding level for a learner (Table 9.2). It may be useful to note that, since interactive problem solving support is constantly available (as immediate feedback) the level of feedback may not be included in PAS categories.

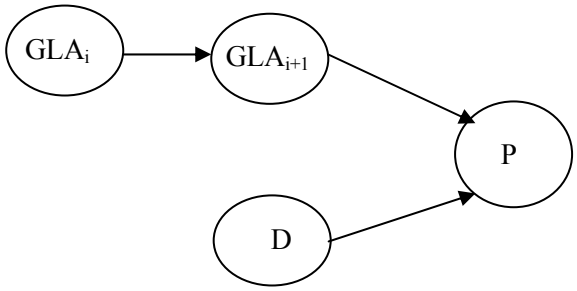


Figure 9.9 Dynamic Causal Relations for Phase-II

GLA	D	P
Strong	Low	Good
Strong	Moderate	Good
Strong	High	Medium
Medium	Low	Medium
Medium	Moderate	Medium
Medium	High	Medium
Weak	Low	Medium
Weak	Moderate	Low
Weak	High	Low

Table 9.1 Fuzzy Rules for Predicting Performance

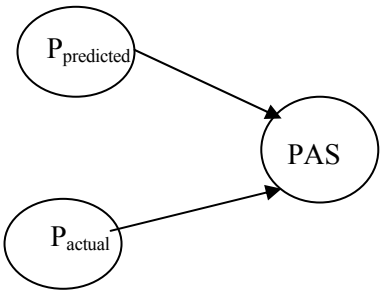


Figure 9.10 Causal Relations for Task Sequencing in Phase-II

$P_{predicted}$	P_{actual}	PAS
Strong	Strong	2 stages up
Strong	Medium	1 stage up
Strong	Low	Same stage
Medium	Strong	1 stage up
Medium	Medium	1 stage up
Medium	Low	Same stage
Low	Strong	1 stage up
Low	Medium	Same stage
Low	Low	1 stage below

Table 9.2 Fuzzy Rules for Task Sequencing in Phase-II

Interactive Problem Solving Support

ANDES (VanLehn *et al.* 2005) is an excellent example for CBL systems that provide interactive problem solving support using BN theory for uncertainty handling. Problem specific causal nets may be created for each testable concept of a case study. Learners' expected behaviour on a testable concept may be associated with one or more mental

states. For example, for the given case study (see Figure 9.4), whether a student could correctly specify the second predicate of the operation schema 'lock' may depend on the strengths of two of his/her mental states: 'Able to use DELTA in operation schemas correctly' and 'Able to use *prime* in operation schemas correctly'. The variable P denotes the performance of a student on a testable concept and takes three values: correct behaviour, not sure, and incorrect behaviour (Figure 9.11). As before, SMS is dynamic. However, since a single variable is involved, BN theory may be used for dynamic prediction. SMS may be updated after each testable concept is identified. A time threshold may be used to avoid senseless predictions.

Instead of prior probabilities, relevant fuzzy rules may be designed for these relationships (Table 9.3). The rules may be more meaningful than fractions assigned for prior probabilities. As stated before, the collection of testable concepts, and the corresponding causal relations and the fuzzy rules may all come from three sources: expert judgment, empirical studies and cognitive task analysis.

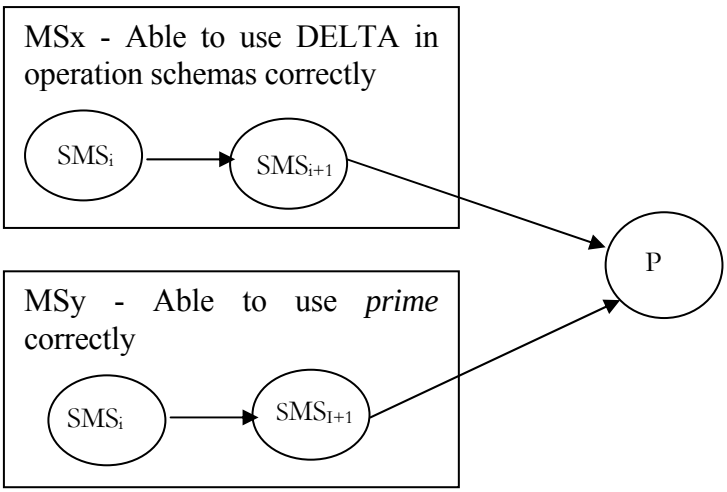


Figure 9.11 Predicting Performance on a Testable Concept

Not only actual performance, but also expected performance may be used to determine the required level of feedback (Table 9.4). The previous causal net used for task sequencing (Table 9.2) may be assumed for predicting PAS during problem solving also. However, the meaning of the variables P and PAS are slightly different in this case. Unlike before, P may be easily defined for each testable concept, and PAS does not include any task sequencing decisions.

MSx	MSy	P
Strong	Strong	Correct
Strong	Medium	Correct
Strong	Weak	Not sure
Medium	Strong	Correct
Medium	Medium	Not sure
Medium	Weak	Wrong
Weak	Strong	Not sure
Weak	Medium	Wrong
Weak	Weak	Wrong

Tables 9.3 Fuzzy Rules for Interactive Problem Solving Support

$P_{redicted}$	P_{actual}	PAS
Correct	Correct	Affirm-Level1
Correct	Not sure	-
Correct	Wrong	Hint-Level1
Not sure	Correct	Affirm-Level2
Not sure	Not sure	-
Not sure	Wrong	Hint-Level2
Wrong	Correct	Affirm-Level3
Wrong	Not sure	-
Wrong	Wrong	Hint-Level3

Tables 9.4 Fuzzy Rules for Feedback during Problem Solving

If the system identifies a correct behaviour on a testable concept, it may give positive feedback using an assertion. The level of information included in this assertion may vary. The level-1 assertion may just inform the learner that s/he is correct, whereas the level-3 may activate the Refinement unit and encourage them to execute the resultant code and see the results. Level-3 assertion is given for a learner who has very low level strength in both mental states, and yet shown the correct behaviour on the testable concept. Therefore, system assumes that executing the code and testing the results would enhance their confidence.

Similarly for a wrong behaviour, the level 1 hint may just inform the learner that s/he is not correct; the level 2 hint may encourage the learner to run the relevant code in Refinement unit and to compare the results with expected results, or simply compare both

codes against the given specification (Figure 9.11), whereas the level 3 hint may be a 'bottom-out hint' (it tells the student explicitly what to do (Conati *et al.* 2002)).

Similar to phase-I, the SMS values may also be updated based on the actual behaviour as evidence. Separate fuzzy rules may be defined in the reverse direction (that should go well with previous tables). Instead, BN theory may be used.

9.2.4 Refinement Unit

The Refinement unit provides an interactive environment for systematically transforming Object-Z construct to Java code. In phase-I, the Refinement unit includes only a mapping and a matching process. It maps a given specification to one or more Java code. In phase-II, however, since the learner may create their own specifications, it should be able to automatically convert specifications to code (Figure 9.12).

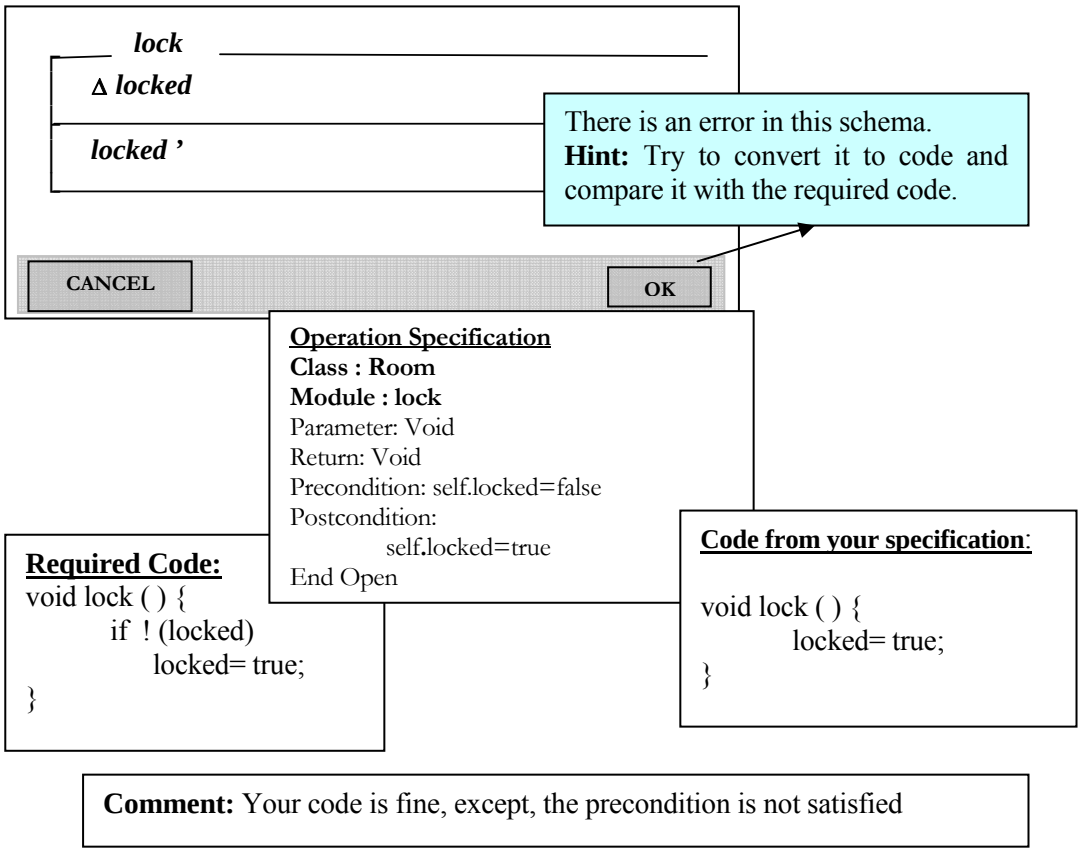


Figure 9.12 Interactive Help on Semantics

During the early stages, the students may be asked just to compare the code automatically refined from their own specification with the corresponding code stored in the system for

the given specification. At the later stages, the students may be asked to gradually convert their own specifications to code; in this case, the Refinement unit should be able to guide learners through the transformation process (Figure 9.12). This feature encourages the learner to engage in some meaningful constructional activities through problem transformation.

9.3 Transforming UML models to Object-Z

9.3.1 Overview

Problem transformation is used in this research for providing facilities for active learning. In the second phase, where the case study problems require critical responses from students (unlike the earlier phase, where MC tests are used), there are enormous opportunities for constructive learning. At the initial stages, the UML models for both structural and functional abstractions of a case study will be readily available in the system. Therefore, the transformation process can also be included beforehand. However, the learner should be guided through a simple step-by-step algorithm during the process. Moreover, in the later stages, learners may create their own UML models. Therefore, if required, the system should be able to demonstrate a simple algorithm for the transformation process. Although, considerable research has been performed towards transforming UML to Object-Z models, none of them attempts to design a suitable algorithm for the translation processes. A simple step-by-step algorithm is proposed in this research for this purpose. It will be described in detail in this section.

Researchers naturally fall into two groups depending on their ideology about the nature of a specification. The first group, consisting of purists, insist that a specification should purely specify some feature (say requirements) and not necessarily be implementable (Hayes *et al.* 1999). The researchers in the second group maintain that a specification is preferable if it can be further processed (Fuchs 1999). Ultimately, they argue, the best specification is the one that could be refinable to implementation. With the introduction of reference semantics in place of value semantics, the Object-Z notation is more inclined to fit in with the approach of the second group. In this research, in addition to specifications of a declarative nature, some temporal and/or procedural information will be provided using external links in order to make it refinable. The next section will describe the

approach taken in this research for transforming structural aspects of a UML class diagram. Transforming methods to operation schemas will be discussed later.

9.3.2 Transforming Structure of a Class Diagram to a Class Schema

As stated before, there is some key research that attempts to automatically transform the structural part of the class diagrams to Z family notations. Examples of such systems include Dupuy *et al.*'s RoZ (2000b), Miao *et al*.'s OZRose (2002) and Sun *et al.*'s UML to TCOZ (2001). To the author's knowledge, none of them describe how operation schemas could be transformed. Expecting students to express preconditions, post conditions and semantics of UML operation specifications in Z notation is not practical. It is interesting to note that the experts in this discipline are of the opinion that any such automatic transformation could not be completed without some form of human interaction.

In particular, there are some gaps to be filled in existing transformation processes. Some features in UML notation are not available in Object-Z (e.g. protected visibility), and vice versa (e.g. class invariant) – but this issue will not be a problem for this research. Moreover, since the degree of formalism in both notations varies, there are certain issues that should be resolved during transformation. Some of these issues are discussed in detail in the following paragraphs.

There is no protected visibility in Object-Z, and therefore, protected attributes in UML diagrams will be considered private. But, private features cannot be accessed even by the sub-classes. Therefore, relevant public methods will be included to manipulate each such protected attribute. Not only sub-classes, but also all other classes can utilize this attribute. This solution is highly modular, as the attribute can only be accessed through the interface of the class. Nevertheless, protected methods in UML notation can only be made public in Object-Z; there is no other way to make them available for the sub-classes. For each class, a meta class (a class of class - which will have only one instance) will be maintained. All of this missing information can be included in these meta classes.

Object-Z has a way to represent constants. None of the operations can change them. However, these constant values are usually initialized while constructing an object.

Object-Z assumes that the objects are always available in the environment, and therefore, construction and destruction of objects is not possible. A system class (which will have only one instance) will be maintained for including and excluding objects to and from the system.

In Object-Z, there is no way to distinguish between class-level and object-level members. Therefore, the class-level members should also be considered as object-level members. Alternatively, all the class-level members may be included in the meta class of that class.

Overloading is not possible within an object schema. If an attribute and method have the same name, they are slightly changed. For overloaded methods, if any, appropriate names will be selected as operation schema names. Signature compatibility checking will be carried out for every related change.

Moreover, in Object-Z, specific attribute types such as CHAR are not available. Appropriate broad types should be used instead. Any missing information on types could be maintained in the meta class. Attributes are straightaway included in the declaration part of the state scheme. Secondary (or dependant) attributes are separated using delta symbol. Best types are determined for each attribute (if necessary, the learner is asked to provide the best option). The class invariants for each class, if any, should be explicitly given with the UML model, or the learner will be asked to give the class invariants (if they are included in UML as natural language descriptions). They will be included in the predicate part.

Note that INIT is a dedicated schema; it is not an operation but just a condition. The initial values for each attribute are converted to appropriate invariants.

While converting association relationship, whenever possible, Kim *et al*'s (2000a) recursive approach will be used. The appropriate attributes will be included in the classes of both association ends - the corresponding role names may be used to name them. Depending on the multiplicity constraint, it may be designed as a collection of identities or a single identity. If the stereotype of the link is given as "ordered", it should be a sequence. The resultant invariants will be included in both state schemas recursively. The existential constraints may be included as invariants. Instead of depending on the system

schema for manipulating association links a separate meta schema will be included for each association. This will keep the links explicitly. This approach is redundant, but versatile.

If the navigational direction is only one sided, the relevant attributes will be included only into the class opposite to the arrow-head. The reverse link will be lost. Another drawback to this approach is much auxiliary information (such as association names) will be lost in the early stage of the development process. To overcome this, all of that auxiliary information will be included in the meta-class for association.

Aggregation will be considered as a special form of association. The navigation direction towards aggregation will be available always. In general, the parts to whole navigation will also be maintained. Moreover, a separate Boolean type attribute will be used to explicitly distinguish aggregations with similar associations. In addition to the properties of aggregation, a composition has additional constraint. A part object cannot be available in the system without its related whole. For example, if a book is withdrawn from a library, all of its copies should be removed from the system. Like UML, Object-Z also has different notation for this construct. The entire object (not the reference) may be contained in another object. This allows us to avoid additional constraint to specify this unique feature.

An association class and the related association links are handled like as associations. The references of the object on both side of the link are included as attributes, and relevant invariants are included recursively. A separate association schema will be included to represent the association class.

At the specification stage, inheritance is mainly used for clarity (as a general-special relation). Attributes and methods can be reused. Similar to UML, Object-Z provides a special notation and semantics for inheritance. Therefore, the clarity and flexibility of a UML inheritance hierarchy can be transformed to relevant Object-Z schemas. However, it is important to note that, any inherited method in Object-Z will not be overridden as in UML, but, will be conjoined with the corresponding method of the sub-class.

As with UML, Object-Z also supports typed polymorphism. A special symbol is used to declare that an attribute can be an object of a portion of a class hierarchy.

Objects are not stand-alone entities. They communicate with each other in the environment. The environment is determined by the scope of the problem. Objects are considered to be always available in the environment. They will be in their initial state if they haven't undergone any transitions. They can be added to the database or removed from the database. Even after they are removed from the database they will be available in the environment.

The Object-Z class schema is a step ahead of its corresponding UML class with regard to encapsulation – it contains information about not only state and behaviour but also its relationship with other objects. The semantics used in Object-Z for object identity, object reference and self are all similar to corresponding UML semantics. Other complicated approaches, if necessary, will be compared as they are discussed.

The declaration part of operation schemas may be generated automatically. The input (and output) variables may be included in the Object-Z schema. Special treatments may be necessary for alternative and error conditions. However, completing the prediction part may be challenging. The next section will describe this issue in detail.

9.3.3 Transforming Methods to Operation schemas.

Object-Z is a declarative notation. Operation specification in UML may also be given in a declarative form (as a contract using Object Constraint Language). Transforming operation by operation to relevant operation schema may not be effective; and may not be sufficient. Compared to UML, specifying the required behaviour related to error conditions and alternate actions in Object-Z notation is difficult. Moreover, UML methods include message passing. In order to specify a use case realization, the sequence diagram indicates the required functional behaviour of objects in time sequence. The predicates are logical expressions, and therefore, special constructs such as parallel operator (Duke *et al.* 2000) are required to declare sequential behaviour in Object-Z. Transforming methods to operation schemas is not straightforward. Therefore, a step-by-

step methodology is proposed in this research to transform UML class diagram (including methods) to Object-Z schema.

Generally, in OO, attributes are declared as private; however, relevant base operations (public methods such as *set* and *get*) are usually added to manipulate them. In Object-Z schemas, attributes are usually left as public. Moreover, it is not cost effective to create formal specifications for each and every functionality required. However, base operations like *set* and *get* can be easily created automatically. Nevertheless, it is important to create unambiguous specifications, at least for critical requirements. In UML, the use cases are usually ranked according to their importance.

The terms *independent operation* and *cluster* used in the above algorithm description are borrowed from the OO testing discipline and used here with the same semantics (Sommerville 2001). A cluster is a group of operations that are used to perform a particular request. It may contain only a single operation. An independent operation is an operation that does not use operations of any other classes. The proposed algorithm described above follows a cluster-based approach[8]. Clusters are incrementally developed, and as they are completed the learners' level of confidence and mental satisfaction increases. Initially all the independent operations are specified and then the dependent operations that make use of them are incrementally specified. This approach has another advantage: test cases can be easily derived; therefore, stubs and drivers are not necessary.

First, a simple class diagram is considered. There is no inheritance hierarchy. There are only association relationships. Assume use cases are ranked based on their importance.

Step-1: REPEAT steps 2 through 14, UNTIL there are no more important use cases
    Step-2: Select the next important use case (initially select the most important use case)
    Step-3: Consider the corresponding sequence diagram
    Step-4: REPEAT steps 5 through 10, UNTIL there are no more clusters in the sequence diagram
        Step-5 Select the next important cluster (initially select the most important cluster)

[8] Referring to functionality, OO modeling is not hierarchical; therefore, a top-down, or a bottom-up approach is not possible.

Step-6: REPEAT steps 7 through 9, UNTIL there are no more independent operations in the cluster

Step-7: Select the next independent operation in the cluster (initially select the most important operation)

Step-8: Generate the relevant class schema, if not generated before

Step-8.1: Complete the state schema (interact with user, if necessary)

Step-8.1.1: Specify attributes and types

Step-8.1.2: Specify dependent attributes

Step-8.1.2: Specify class invariant

Step-8.2: Complete the INIT schema (interact with user, if necessary)

Step- 8.2.1: Specify initial values for attributes

Step-9: Specify the operation schema (interact with user, if necessary)

Step-9.1: REPEAT UNTIL there are no sub-operations

Step-9.1.1: Select the next independent sub-operation

Step-9.1.2: Check the name, delta list, input and output variables

Step-9.1.3: Complete the declaration part

Step-9.1.4: Complete the prediction part (this is challenging)

Step-9.2 Complete the operation schema

Step-10: REPEAT Steps 11 through 13, UNTIL there are no more operations in the cluster.

Step-11: Incrementally select the next important operation that depends only on the operations that are already considered

Step-12: REPEAT UNTIL there are no more associations left

Step-12.1: Select the next important association

Step-12.2: Generate the class schema at the other end (similar to Step-8)

Step-12.3: Include relevant attributes into both state schemas

Step-12.4: Recursively include relevant invariants into both state schemas

Step-13: Specify the operation specification (similar to Step- 9)

Step 14: Complete Operation Schemas

Step-15: Consider alternative sequence diagrams (if any).

Even if there is no such diagram, alternate trivial use cases may be available. Check the use case specification and the relevant operation specification. If so, modify the schemas accordingly.

Step-16: Refine the schema

Illustration

The case study described in Chapter 4 will be used here for illustration. There are subtle changes made in the diagram. For example, the operation 'roomOK' is explicitly shown in the relevant sequence and class diagrams (however the'roomNotLocked' case is not considered).

Step 1: Repeat steps 2 through 14, Until there are no use cases left

Step 2: Suppose the use case *open room* is the most important (Figure 9.1)

Step 3: The sequence diagram for the main use case is selected (Figure 9.13)

Step 4: Repeat steps 5 through 10, Until there are no more clusters

Step 5: The important cluster operation is selected

(here it is *Key.open( )* – see Figure 9.13)

Step 6: Repeat steps 7 through 9; until there is no independent operation

Step 7: The next important independent operation is selected

(here it is *Room.open( )* – see Figure 9.13)

Step 8: Generate the class schema for the *Room* class. It will also include a skeleton for the operation *open( )* (Figure 9.14).

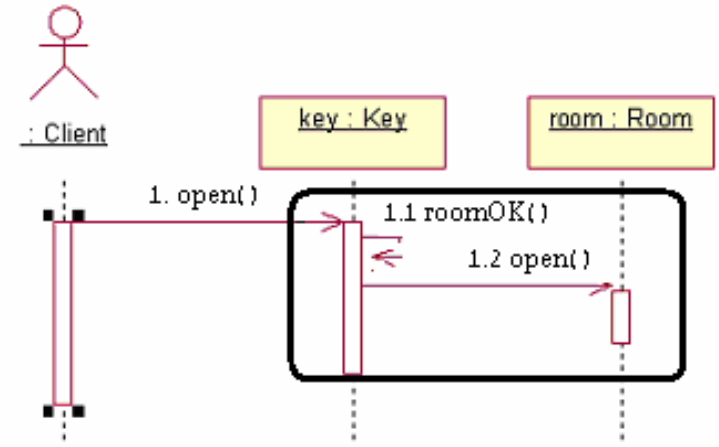


Figure 9.13 Sequence Diagram for the Main Usecase

Step 9: Based on the operation specification (Figure 4.3), complete the operation schema open in *Room* class (Figure 9.15). It will be relatively easy, if the pre and post conditions were given in some formal notations.

Step 10: Repeat steps 11, 12 and 13 until there are no more operations left

Step 11: Select the operation Key.open( ), which uses Room.open( ) operation (Figure 9.14).

Step 12: The association 'open' provides link for this message passing (see Figure 9.15).

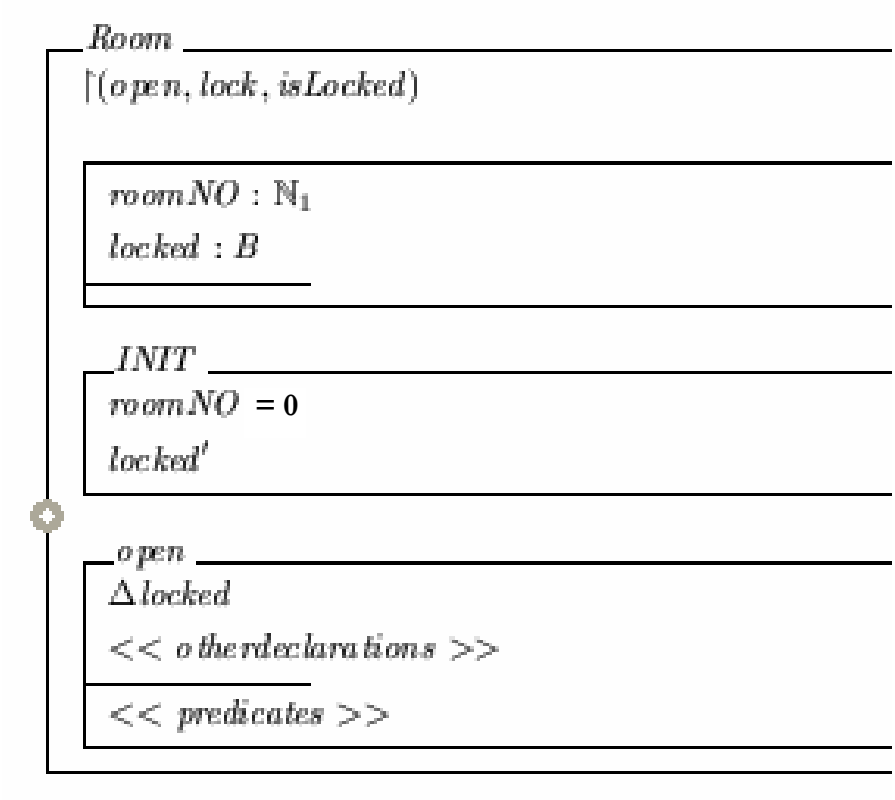


Figure 9.14 Class Schema Generated for Room

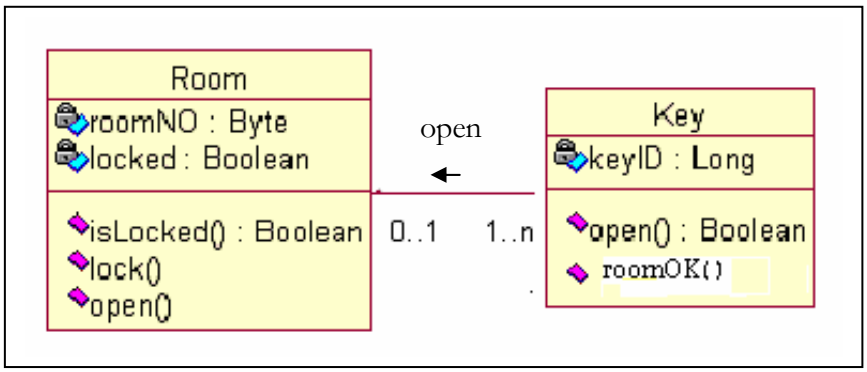


Figure 9.15 Class Diagram

Step 12.1: The Key class is at the other end of the association ‘open’ (Figure 9.15). Therefore, the class schema for Key will be generated. It will also include a skeleton for the operation ‘roomOK’.

Step 12.2 & 12.3: The association is specified now. It will be considered as bidirectional. The relevant attributes and invariants will be included recursively into the state schemas of both participating classes (Figure 9.16).

Step 13: The operation open of Key class is to be specified. This is similar to Step 8.

Step 10: Repeat steps 11, 12 and 13 (there are more operations to be specified)

Step 11: Independent Sub-operation ‘roomOK’ is to be specified (Figure 9.15)

Step 12: No new association

Step 13: Use the given operation specification (Figure 4.3) to complete the ’roomOK’ schema

Step14: Complete the operation ‘open’ (Figure 9.17: use the corresponding operation specification)

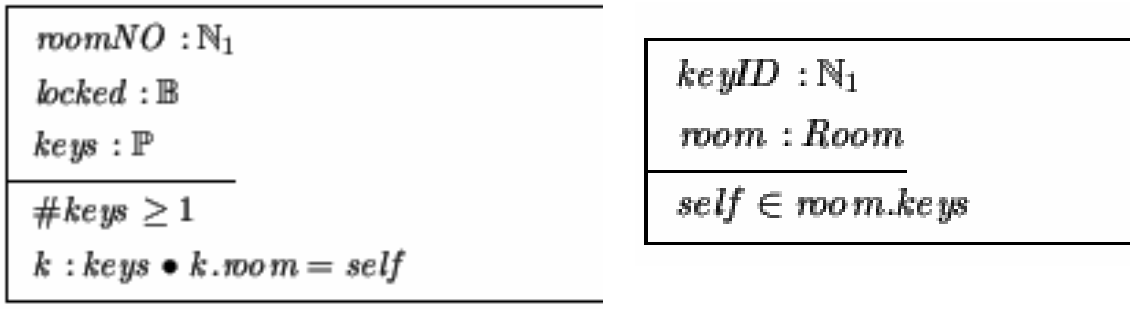


Figure 9.16 State Schemas of Room & Key

Figure 9.17 Operation Partially Completed

Step 15: The alternative sequence diagram is not available. But alternate use case is available. Check the use case specification and the relevant operation specification. Modify the schemas.

According to the use case specification (Figure 9.1), the Room may not be opened by the given Key. Therefore, an alternate schema is required to cover this case. The operation roomNotOk( ) is designed for this purpose (Figure 9.18). It is a straightforward operation. Later the operation open (Figure 9.17) will be modified to match the specification in Figure 9.1 (see Figure 9.18).

In the above illustration some peripheral class schemas are not included: for example, meta classes, system class, and association (meta) class. In fact, they are not important at this stage.

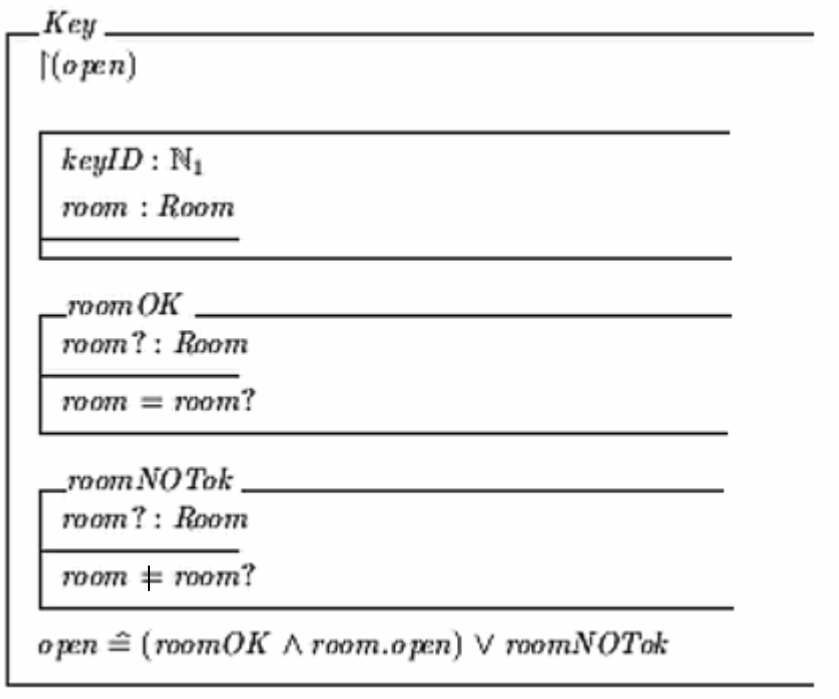


Figure 9.18 Class Schema for Key

Alternate Designs

At the later scaffolding stages, learners are allowed to devise their own abstraction models. For example, for the given case study, if the navigational direction is not given, a learner may assign the whole responsibility of the use case ‘opening a room’ (Figure 4.3) to the corresponding room object. Therefore, the Room class should include operations such as ‘keyOK’ to check the given key object can open itself (Figure 9.18 & 9.19). This is possible since the Room class now include an attribute to hold the set of all of its key objects (actually they are references of objects – since the relation is not composition).

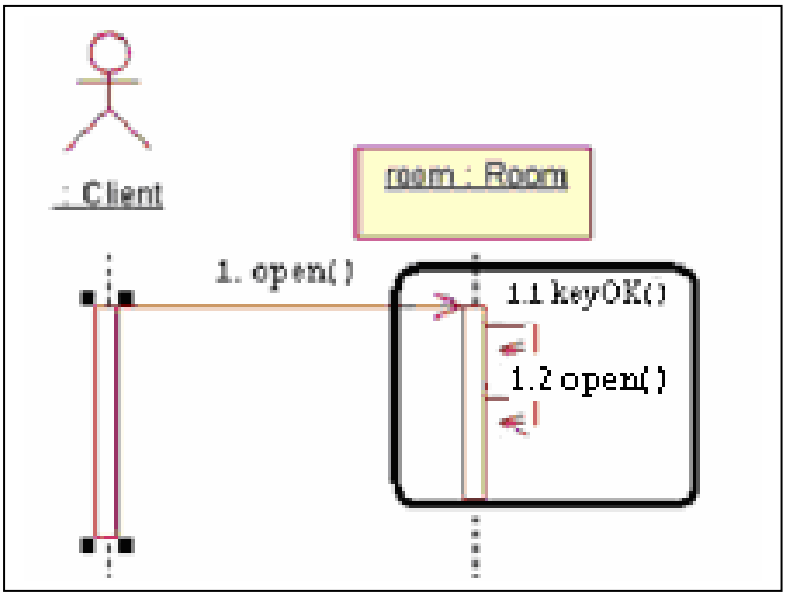


Figure 9.19 Alternate Use case Realization: Sequence Diagram

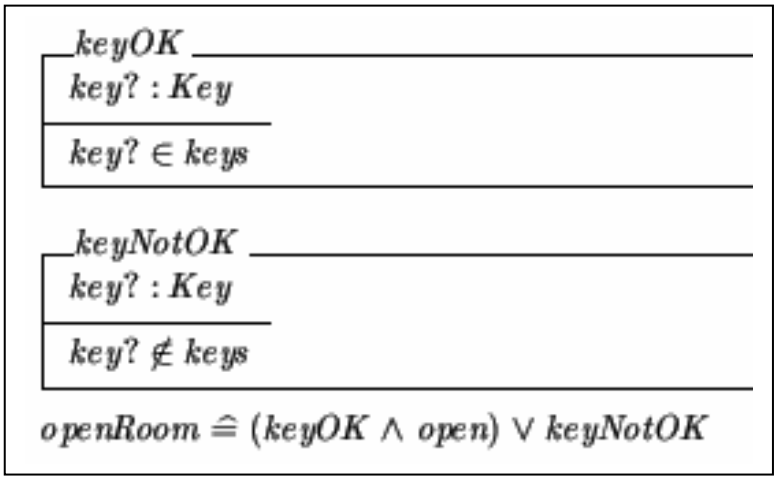


Figure 9.20 Operations of an Alternate Schema

Some Complex Cases

There may be many ways to solve a case study problem. There may be many common mistakes identified. For potential misconceptions, the relevant feedback is formulated and recorded. The system should be able to provide interactive help for different approaches.

Case 1: The selected use case may have dependent (include or extend) use cases

In this situation, the dependent use case should be considered first.

Case 2: The selected use case may be a generalized use case.

The special use cases, if any, should be considered first.

Case 3: A class may be a subclass: In particular the current operation is an inherited operation.

First consider the ancestor class that first specifies the current operation, and continue downwar

Case 4: There may be association classes.

The association schema will be defined after specifying corresponding association.

Case 5: There may be aggregate or composite relationships.

Aggregation will be considered as association (except for an additional tag to keep the difference). There is a special construct in Object-Z for composition.

9.4 Summary

Learning the syntax and semantics of the Object-Z notation alone is usually not sufficient for creating efficient specification from textual descriptions Similar to piloting, learners need to practice the skills in real (or simulated) settings. A constructive learning environment is designed in the second phase for the users to get hands-on experience in creating Object-Z specification. Learners will be given case studies in textual description with relevant UML models, and asked to produce Object-Z specification. An instructional strategy based on three levels of scaffolding is proposed in this research to deal with abstraction problem. The case studies at the first level provide data and functional abstraction; at the second level they provide only the functional abstractions. Finally, at the third level, the learners are encouraged to perform data and functional abstractions by themselves.

Within each level, a number of scaffolding stages are designed to provide support on structural aspects of the notation. Initially, almost the entire structure of a class schema will be provided, and the learners only requested to fill-in a place holder. The detail to be filled-in may be a single predicate at the start, and eventually, in the final stages, it may become the whole class schema. Learners are initially expected to have sufficient knowledge of the syntax and semantics of Object-Z constructs, and ultimately, as they

progress, they should be able to create specification document using Object-Z notation for real world problems.

At the first two levels, the learners are encouraged to transform UML models to Object-Z notation. This feature engages learners in some meaningful constructive activity, and this in turn helps them to build their Object-Z skills effectively. Nevertheless, since the translation process is significantly challenging, the system should be able to guide learners in this process systematically. In this research, a step-by-step methodology for transforming simple UML class diagrams to equivalent Object-Z schema is proposed. This methodology is based on use case-realization process and requires relevant human interaction for its success (in fact, except for most trivial cases, it is not possible to transform UML models to Object-Z completely automatically). A simple case study is used in this chapter to illustrate the methodology. Finally, the proposed methodology may be extended for more general cases, and there are some suggestions included for this purpose.

Chapter 10

Conclusions and Future Research

10.1 Conclusions

Some disciplines are inherently complex and therefore challenging to learn. Research in this dissertation shows problem transformation techniques can be effectively used for designing CBL systems for certain complex domains. A suitable instructional strategy is essential to realize this goal. Problem transformation, in general, requires learners to actively construct more complex artefacts based on less complex ones. Moreover, constructivism claims that new knowledge is built on learners' previous knowledge. In particular, revising necessary previous knowledge is essential for problem transformation. For example, knowledge in differentiation is fundamental for learning integration. Related to the exemplar domain of this research, UML is advantageous for learning Object-Z. Transforming UML models to Object-Z models is easier than creating Object-Z models from scratch. Furthermore, scaffolding can be used to alleviate the difficulty associated with complex transformation processes at the initial learning stages. This research also demonstrates that an instructional model that includes explicit phases for pre conditioning and scaffolding (modelling and building) is suitable to realize problem transformation efficiently. The exploration phase in this model encourages learners to critically analyse their stance often and update their beliefs accordingly.

A Learner model provides adaptive help tailored to the individual learner, and therefore, plays an important role in CBL systems. A simple, locally intelligent Learner model is used in this research. Uncertainty is inevitable in Learner models. Research shows that BBN can be efficiently used to handle uncertainty. This research asserts that using fuzzy logic based strategy for uncertainty management in Learner model is easy to implement as well as efficient. The fuzzy rules used in this model for pedagogical action selection can be formulated to match decision making processes of typical educators. Compared to BBN based models, implementing a fuzzy Learner model is uncomplicated.

The main cause for uncertainty in the Learner model is insufficient bandwidth for evidence. Opening the Learner model provides channels for learners to view and alter their own model. This research illustrates that opening a fuzzy logic based Learner model is both effortless and effectual. Since the associated rules are simple and resemble human reasoning processes, the fuzzy model is easy to understand. A subjective evaluation of the model was conducted.

Formative assessment is essential for scaffolding. MC tests can be used for this purpose. However, the traditional MC test format is not suitable to measure learners' knowledge level accurately. Identifying no knowledge or partial knowledge is essential for the system to give suitable feedback. There is no way to measure partial knowledge in the traditional format. This research maintains that the proposed CBM schema measures the knowledge state of a learner efficiently, and therefore, the system is able to provide suitable feedback. Subjective evaluation was conducted.

In this research, Object-Z is used as exemplar domain. In addition to the general findings described above, some issues pertinent to this domain are worth mentioning here. Motivation is essential for learning. Graduate students are extraneously motivated if they feel that learning a particular domain may be beneficial in long term. The Refinement unit provides such opportunity for the students to realize that learning complex Object-Z notation is really useful for software development. In addition, the Refinement unit will also provide more active learning opportunities by engaging students in the process of transforming formal notation to programming code.

Finally, for problem transformation, students are encouraged to translate specifications in UML to Object-Z notation. To the knowledge of the author, there is no step-by-step methodology available to facilitate this translation process. This research illustrated that the proposed step-by-step methodology to transform UML Models to Object-Z specification is suitable for pedagogical purposes. This approach is complete for simple UML class diagrams without inheritance hierarchies. Other than pedagogical purposes, this methodology can be used to implement semi-automated translation systems for UML to Object Models.

10.2 Contributions

The benefits of this research are outlined below.

- First and foremost, the overall benefit is the design artefacts of a CBL system for certain complex disciplines. In particular, the instructional strategy used in this design is based on the FOPSI model and incorporates necessary techniques to facilitate active learning using problem transformation. The FoPSI model is based on the constructivist learning theory and includes four phases: Pre-Conditioning, Modelling, Building, and Exploring. During pre-conditioning, learners refresh their past knowledge relevant to the current learning concept, and then build their own understanding of the concept in the next two phases. The FoPSI model helps to alleviate complexity associated with learning certain disciplines. In the building phase, learners could initially focus on less complex issues, and gradually (and systematically) move to more complex issues. The exploration phase encourages a learner to engage in meta-cognitive activities.

- The most concrete contribution is the proposed fuzzy-logic based strategy for managing uncertainty in Learner models. The underlying rules for this model are simple and can be related to expert teachers' procedures. Using simple BBN and Dynamic BBN to complement fuzzy model is another novel and useful idea.

- An approach for opening the fuzzy-logic based Learner model (and also opening the mentoring strategy) is presented. The advantages and disadvantages of opening such models are also analysed.

- A simple step-by-step methodology is used to transform UML notation to Object-Z notation. Though the methodology is originally designed for pedagogical purposes, it can be readily used to implement semi-automated translating system.

- A novel CBM scheme is designed in this research. This scheme is better than traditional MC tests, because it can measure students' confidence level also on each option. This scheme can be easily implemented.

- The prototype of LOZ developed in this study is another tangible benefit of this research. This prototype was constructed in evolutionary fashion and includes many reusable components. A fully functional CBL system may be implemented by extending (and modifying) this prototype.

- A detailed design for phase- II of the CBL system for learning Object-Z is given in Chapter 9. This design can be used as a basis for implementing a simple home-work practice environment.

10.3 Future Research

A number of research problems that can arise out of this research are discussed below. The first item describes what others could directly take form this research. The next two issues may lead to major research projects, while the rest may still demand significant research initiatives.

- Designing Problem Transformation Based CBL systems

In this research, an instructional framework that uses problem transformation as its underlying instructional strategy has been designed. This framework includes a four-phase instructional model to support active learning through scaffolding. How can this frame work be utilized for designing other complex domains? Complexity is caused by the short term memory demand associated with learning a discipline due to various reasons such as computational difficulties, complex coupling between entities etc. For example, in high schools, integration may be taught by using differentiation. While solving the UV form of integration, students may be initially supported by relevant differentiation results, until they become comfortable with splitting complex structures into simple constituents. The subject domain should be analysed and the framework modified slightly to accommodate domain specific properties. Similarly, a new object oriented language may be taught using a known imperative language. The system may provide partial solutions for relevant methods and allow the student to complete the rest whilst concentrating on learning object oriented concepts.

- Fuzzy Learner Model for Interactive Problem Solving Support

In this research, for the phase-I of the CBL system, a fuzzy Learner model is used for task sequencing and feedback. Learner models can also provide adaptive interactive problem support. ANDES-I (VanLehn *et al.* 2005) is an example of a CBL system that provides interactive problem solving support under uncertainty using BN theory[10]. Instead of probability tables, relevant fuzzy rules could be formed after a cognitive task analysis. In Chapter 9, the idea of incorporating a fuzzy Learner model for interactive problem solving support is suggested.

The fuzzy rules, as they could reflect human reasoning process, would be more meaningful and straightforward. The system could provide intelligent interactive help during the problem solving process based on the current state of the solution and learning ability of the particular student. Moreover, the underlying fuzzy rules may be improved using machine learning techniques. As many people use the system, it would become more accurate. This line of investigation would be innovative and challenging.

- Fuzzy Utility Function for Task Sequencing

Traditionally, Learner models are used for task sequencing (next problem or next topic selection) and/or interactive problem solving. The tutoring decisions are usually not explicit but combined with Learner model logic. Mostly they are ad hoc and hardly based on any standard theory. In this research, the tutoring decisions are included in the fuzzy rules. Nevertheless, there is some research that uses statistical decision theory based on utility functions for post-diagnostic tutoring decisions (Reye 1995) in order to optimize the learning outcome of students. It would be interesting to dissect the fuzzy rules used in this research, and separate the tutoring decision making process from the Learner modelling aspects. Is it beneficial to use a fuzzy utility function along with the fuzzy Learner model to support optimal tutoring decisions? The above question could lead to an innovative research project.

10 However, later in ANDES-II, the BN Learner model was abandoned and instead an unintelligent help unit is included (Van Lehn *et al*, 2005). Task sequencing support is not necessary in ANDES, because it is not designed for a self-paced course and also it need not to select next assessment problem or next topic for students, and therefore, the authors argue, BN Learner model, though efficient, is not necessary.

- Opening a Fuzzy Learner Model

The idea of opening the Learner model is presented in this research. In fact, a small portion of the Learner model is made open, and a subjective evaluation was conducted to evaluate the effect. Due to time limitation, however, an objective evaluation was not conducted. Which problems could we encounter when opening the entire Learner model and mentoring steps? What characteristics of the fuzzy model would favour opening? And which would have a negative effect? Moreover, will that be effective? If not, why not? Conducting research in this direction would be interesting.

- CBM Scheme

The idea of utilizing a confidence based MC test scheme for formative assessment was suggested in this research. This scheme can measure partial knowledge effectively, and in turn, could help the system to compose suitable feedback. Though the subjective evaluation confirms this claim, objective evaluation is common in this type of research. Designing a suitable interface is a challenging task. Are the test items okay? Does the interface help the objective? Will it really measure partial knowledge? Can it be compared with a related critical response test? if so, how? These questions will lead to a useful line of analysis.

- Using Fuzzy Logic for Web-based Learner Model

The phase-I of the design was successful. Will it be possible to port this design to a web-based environment? What are the new issues? Which is the suitable location for the Learner model, client or server machine? How can fuzzy logic be used for adaptable hypermedia? This line of investigation could be suitable for a small project.

- UML to Object-Z translation: A semi-automatic approach

A step-by-step methodology for translating basic UML class diagrams to Object-Z schemas is proposed in this research. However, inheritance hierarchies are very common in class diagrams. The above methodology can be extended to class diagrams with inheritance hierarchies. Translating a complex class diagram to Object-Z schemas is impossible without human intervention. Effectiveness of the methodology is to be thoroughly tested.

References

.

Ackermann, E. (2004). Constructing Knowledge and Transforming the World. A learning zone of one's own: Sharing representations and flow in collaborative learning environments. M. Tokoro and L. Steels. Amsterdam, Berlin, Oxford, Tokyo, Washington DC, IOS Press, 2004.**:** pp. 15-37.

Ackermann, E. (2007). Experiences of artefacts: People's appropriation, objects' affordances. Key works on radical constructivism. M. Larochelle. NL, Sense Publishers**:** pp. 149-159.

ACM-SE (2004). "Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering." A Volume of the Computing Curricula Series: 59.

Adam, A. and J. Laurent (1980). "A system to debug student progress." Artificial Intelligence **15**: 75-122.

Ainsworth, S. and P. Fleming (2006). "Evaluating authoring tools for teachers as instructional designers." Computers in Human Behavior **22**(1): 131-148.

Albacete, P. L. and K. A. VanLehn (2000). Evaluating the Effectiveness of Cognitive Tutor for Fundamental Physiscs Concepts. Fifth International Conference - ITS'2000, Montreal, Canada.

Amalio, N. and F. Polack (2003). Analysis and Comparison of Formalization Approaches of UML class constructs in Z and Object-Z. ZB 20003, Turku, Finland., LNCS, Springer.

Anderson, J. R. (1983). The Architecture of Cognition. Cambridge, MA, Harvard University Press.

Anderson, J. R. (1993). Rules of the Mind. Hillsdale, NJ, Erlbaum.

Anderson, J. R., A. T. Cobertt, K. R. Koedinger and R. Pelletier (1995). "Cognitive Tutors: Lessons Learned." The Journal of Learning Sciences **4**(2): 167-207.

Anderson, J. R., R. Farrell and R. Sauers (1984). "Learning to program in Lisp." Cognitive Science **8**: 87-129.

Anderson, J. R. and K. A. Gluck (2001). What role do cognitive architectures play in intelligent tutoring systems? Cognition and instruction: 25 years of progress. D. Klahr and S. Carver, M. New Jersey, Lawrence Erlbaum.

Anderson, J. R. and R. Jeffries (1985). "The LISP tutor." BYTE **10**(4): 159-175.

Anderson, L. W., D. R. and D. R. Krathwohl (2000). A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon

Araujo, J. (1996). Metamorphosis: An integrated object-oriented requirement analysis and specification method. Department of Computing, University of Lancaster, UK.

Atkinson, R. C. and R. M. Shiffrin (1968). Human memory: A proposed system and its control processes. The Psychology of Learning and Motivation. K. W. Spence and J. T. Spence. London:, Academic Press. **2**.

Azevedo, R. C. and R. M. Bernard (1995). "A meta-analysis of the effects of feedback in computer -based instruction." Journal of Educational Computing Research **61**(2): 213-238.

Barden, R., S. Stepney and D. Cooper (1994). Z in Practice.

Barr, A., M. Beard and R. C. Atkinson (1975). "A rationale and description of a CAI program to teach the BASIC programming language." Instructional Science **4**: 1-31.

Bartlett, F. C. (1932). Remembering: An Experimental and Social Study. Cambridge, Cambridge University Press.

Bednarik, R., A. Moreno, N. Myller and E. Sutinen (2005). Smart Program Visualization Technologies: Planning a Next Step. IEEE International Conference on Advance Learning Technologies(ICALT 2005), Kaohsiung, Taiwan.

Bednarik, R. and M. Tukiainen (2006). An Eye-tracking Methodology for Characterizing Program Comprehension. Symposium on Eye Tracking Research and Applications, ETRA 2006, San Diego, CA, USA, ACM Press.

Bell, M., A. and D. Jackson (1993). CALVIN Courseware Authoring Language using Visual Notation. IEEE Symposium on Visual Languages, Bergen, Norway.

Ben-Simon, A., D. V. Budescu and B. Nevo (1997). "A comparative study of measures of partial knowledge in multiple-choice tests." Applied Psychological Measurement **21**(1): 65-88.

Bennett, F. (2000). Computers as Tutors: Solving the Crisis in Education, FABEN, INC.

Berdie, D. R., J. F. Anderson and M. A. Niebuhr (1986). Questionnaires : design and use Metuchen, N.J., Scarecrow Press.

Bergin, R. A. and U. G. H. Fors (2003). "Interactive simulated patient - an advanced tool for student-activated learning in medicine and healthcare." Computers & Education **40**: 361-376.

Berkin, R. C. and S. Trubatch, L. (1997). Fuzzy Systems Design Principles. New York, IEEE Press.

Bhuiyan, S., J. Greer and G. McCalla (1994). " Supporting the learning of recursive problem solving." Interactive Learning Environments **4**(2): pp. 115-139.

Bitzer, D., P. Braunfeld and W. Lichtenberger (1961). "PLATO: An automatic teaching device." IRE Trans. Educ **E**(4): 157-161.

Blair, A., N and M. Ayyub, B (1999). Fuzzy Stochastic Cost and Schedule Risk Analysis: MOB Case Study. Third International Workshop on Very Large Floating Structures (VLFS '99), Honolulu, Hawaii.

Bloom, B. S. (1956). Taxonomy of educational objectives; the classification of educational goals. New York,, Longmans Green.

Board, B. (2000). "http://www.blackboard.com/us/index.Bb (accessed on 20/05/2007)."

Bonar, J. and R. Cunningham (1986). BRIDGE: An intelligent tutor for thinking about programming. ICAI Research Workshop, Windermere, Cumbria.

Booch, G., J. Rumbaugh and I. Jacobson (1999). The UML User Guide, Addison-Wesley.

Bowen, J., P., Ed. (1994). Z User Group (www.zuser.org - accessed in May, 2006). Cambridge, UK, Springer/BCS.

Bowen, J. P. (2000). Experience in Teaching Z with Tool and Web Support. London, Centre for Applied Formal Methods, South Bank University.

Bradbard, D. A., D. F. Parker and G. L. Stone (2004). "An alternative Multiple-Choice Scoring Procedure in a Macroeconomics Course." Decision Sciences Journal of Innovative Education, USA. **2**(1).

Breu, R., U. Hinkel, C. Hofmann, C. Klein, B. Paech, B. Rumpe and V. Thurner (1997). Towards a Formalization of the Unified Modelling Language. ECOOP.

Broberg, A. (1999). Learners as Knowledge Workers: Some Implications. Frontiers in Education, FIE'99, Puerto Rico, San Juan.

Brown, J. S., A. Collins and S. Duguid (1989). "Situated cognition and the culture of learning." Educational Researcher **18**(1): 32-42.

Brown, J. S. and K. VanLehn (1980). "Repair theory: A generative theory of bugs in procedural skills." Cognitive Science **4**(pp. 379-426.).

Brown, M., H. (1987). Algorithm Animation, MIT Press.

Bruel, J., M and P. Chair (1998). Integrating Formal and Informal Specifications Techniques. Why? How? Workshop on Industrial-strength Formal Techniques.

Brusilovsky, P. (1995). Intelligent learning environments for programming: the case for integration and adaptation. AI-ED 95, 7th World Conference on Artificial Intelligence in Education., Washington, DC, USA., Assoc. Advancement of Comput. Educ. Charlottesville, VA, USA.

Brusilovsky, P., E. Schwarz and G. Weber (1996). ELM-ART: an intelligent tutoring system on World Wide Web. Intelligent Tutoring Systems. Third International Conference, ITS '96, Montreal, Canada, Springer-Verlag.

Brusilovsky, P. and J. Vassileva (2003). "Course sequencing techniques for large-scale web-based education." International Journal of Continuing Engineering Education and Lifelong Learning **13**(1-2): 75-94.

Bull, S. (2004). Supporting Learning with Open Learner Models (Keynote Speech). Information and Communication Technologies in Education, Athens.

Burton, R. R. (1982). Diagnosing Bugs in a Simple Procedural Skill. Intelligent Tutoring Systems. D. Sleeman and J. Brown, S. London, Academic Press**:** 236-246.

Bush, M. (2001). "A Multiple-Choice test that rewards partial knowledge." Journal of Further and Higher Education **25**(2): 157-163.

Carbonell, J. R. (1970). "AI in CAI: an artificial intelligence approach to computer-assisted instruction." IEEE Transactions on Man-Machine Systems **11**(4): 190-202.

Chin, N., D (2001). "Empirical Evaluation of User Models and User-Adapted Systems." User Modelling and User-Adapted Interaction **11**: 181-194.

Clancey, W. J. (1986). "From Guidon to Neomycin and Heracles in Twenty Short Lessons." AI Magazine.

Cohen, J. (1988). Statistical power analysis for the behavioral sciences (2nd ed.). Hillsdale, NJ, Lawrence Earlbaum Associates.

Coladarci, T., D. Cobb, C., W. Minium, E. and C. Clarke, R. (2004). Fundamentals of statistical reasoning in education, New York : J. Wiley & Sons, c2004.

Collins, A., J. Brown, S. and S. Newman, E. (1990). Cognitive Apprenticeship: Teaching the Crafts of Reading, Writing and Mathematics. Knowing, Learning and Instruction: Essays on honour of Robert Glaser. L. Resnick, B. Hillsdale, NJ, Lawrence Erlbaum**:** 453-494.

Collins, J. A., J. E. Greer and S. X. Huang (1996). Adaptive Assessment Using Granularity Hierarchies and Bayesian Nets. Proceedings of the Third International Conference on Intelligent Tutoring Systems, Springer-Verlag.

Conati, C., A. Gertner and K. Vanlehn (2002). "Using Bayesian networks to manage uncertainty in student modeling." User Modelling and User-Adapted Interaction **12**(4): 371-417.

Cook, W. (1991). Object-oriented programming versus abstract data types. Foundations of Object-Oriented Languages- Proceedings of REX School/Workshop- LNCS 489. J. de Bakker, W, W. de Rover, P and G. Rozenberg, Springer-Verlag**:** 151-178.

Coombs, C. H., J. E. Milholland and F. B. Womer (1956). "The assessment of partial knowledge." Educational and Psychological Measurement **16**: 13-37.

Cooper, G. F. (1990). "The Computational-Complexity of Probabilistic Inference Using Bayesian Belief Networks." Artificial Intelligence **42**(2-3): 393-405.

Corbett, A., T. and J. Anderson, R. (1992). Student modeling and Mastery Learning in a Computer-Based Programming Tutor. Intelligent Tutoring Systems (ITS'92), ontreal, Canada, Berlin: Springer-Verlag.

Corbett, A., T. and J. Anderson, R. (1993). Student modeling in an intelligent programming tutor. Cognitive Models and Intelligent Environments for Learning Programming. E. Lemut, B. Du Boulay and G. Dettori. Berlin: Springer-Verlag.**:** 135-144.

Craigen, D., S. Gerhart and T. Ralston. (1993). An international survey of industrial applications of formal methods; volume 1: Purpose, approach, analysis and conclusions; volume 2: Case studies. Technical Report NIST GCR 93/626, National Institute of Standards and Technology.

Crocker, D. (2003). Teaching Formal Methods with Perfect Developer. Teaching Formal Methods: Practice and Experience (Workshop), Oxford Brookes University, BCS-FACS.

Crowder, N. A. (1959). Automatic tutoring by means of intrinsic programming. Automatic Teaching: The state of the art. E. Galanter, Wiley.

CTGV (1993). "Anchored instruction and situated cognition revisited." Educational Technology **33**(3): 52- 70.

Cunningham, D., J. (1992). Assessing Constructions and Constructing Assessments: A Dialogue. Constructivism and the Technology of Instruction: A conversation. M. Duffy, T and H. Jonassen, D. Hillsdale, NJ, Lawrence Erlbaum.

Dean, N., C. and M. Hinchey, G. (1996). Formal Methods and Modelling in context. Teaching and Learning Formal Methods. N. Dean, C. and M. Hinchey, G. London, Academic Press Ltd.

Deek, F., P., K. Ho and H. Ramadhan (2001). "A Critical Analysis and Evaluation of Web-Based Environments for Program Development." Journal of Internet and Higher Education **3**(4): 223-269.

Deek, F. P. and J. McHugh (1998). "A Survey and Critical Analysis of Tools for Learning Programming." Journal of Computer Science Education **8**(2): 130-178.

Derry, S., J and L. Hawkes, W (1992). Toward Fuzzy Diagnostic Assessment of Metacognitive Knowledge and Growth. Annual Meeting of the American Educational Research Association, San Francisco, CA.

Devedzic, V., J. Debenham and D. Popvic (2000). "Teaching Formal Languages by an Intelligent Tutoring System." Education Technology & Society **3**(2): 2000.

DeWolf, M. and E. Milgrom (1992). "Tutoring systems for learning to program." Technique et Science Informatiques, vol.11, no.6, 1992, pp.9-37. France. **11**(6): 9-37.

Dick, J. and J. Loubersac (1991). "Integrating Structured and Formal Methods: a visual approach to VDM." Lecture Notes in Computer Science (3 rd International Conference ESEC'91) **550**.

Dick, W. (1991). "An Instructional Designer's View of Constructivism." Educational Technology **31**(5).

Dix, A., J. Finlay, G. Aboud and R. Beale (2007). Human-Computer Interaction, Prentice-Hall. .

Domingos, P. and M. Pazzani (1997). "On the optimality of the simple Bayesian classifier under zero-one loss." Machine Learning: pp 103-130.

Dong, J. S. and R. Duke (1995). "The geometry of object containment." Object-Oriented Systems **2**(1): 41-63.

du Boulay, B. (1988). Intelligent Systems for Teaching Programming. Artificial Intelligence Tools in Education. P. Ercoli and R. Lewis, Elsevier Science (North-Holland).

du Boulay, B. (1992). Towards more versatile tutors for programming. New Directions in Educational Technology, NATO Advanced Research Workshop, Milton Keynes, UK, Springer-Verlag.

du Boulay, B., B. Romero, R. Cox and R. Lutz (2003). Towards a Debugging Tutor for Object-Oriented Environments. AIED'2003, Sydney, Australia.

du Boulay, B. and C. Sothcott (1987). Computers Teaching Programming: An introductory survey of the field. Artificial Intelligence and Education. Lawler and M. Yazdani, Ablex. **1:** 345-72.

Duke, R., K. Paul, R. G. A and S. Graeme (1991). The Object-Z specification language version 1. Technical Report, Software Verification Centre, Department of Computer Science, University of Queensland.

Duke, R. and G. Rose (2000). Formal Object-Oriented Specification Using Object-Z. London, MacMillian.

Dunlop, W. P., J. M. Cortina, J. B. Vaslow and M. J. Burke (1996). "Meta-analysis of experiments with matched groups or repeated measures designs." Psychological Methods **1**: 170-177.

Dupuy, S., Y. Ledru and M. Chabre-Peccoud (1998). Translating the OMT Dynamic Model into Object-Z. ZUM' 98.

Dupuy, S., Y. Ledru and M. Chabre-Peccoud (2000a). Integrating OMT and object-Z. BCS FACS/EROS ROOM Workshop, technical report GR/K67311-2. A. Evans and K. Lano, Imperial College, London.

Dupuy, S., Y. Ledru and M. Chabre-Peccoud (2000b). An overview of RoZ- a tool for integrating UML and Z specifications. 12th Conference on Advanced information Systems Engineering (CAiSE'2000), Stockholm, Sweden.

Eisenstadt, M., B. A. Price and J. Dominque, Eds. (1992). Redressing ITS fallacies via software visualization. Cognitive Models and Intelligent Environments for Learning Programming.

Elsom-Cook, Ed. (1990). Guided Discovery Tutoring: A framework for ICAI research, Chapman, London.

Faust, G. W. (1974). "Design strategy and the TICCIT system." Viewpoints **50**(4): 91-101.

Finney, K. (2003). Hello class- let me introduce you to Mr.Six. Teaching Formal Methods: Practice and Experience (Workshop), Oxford Brookes University, BCS-FACS.

Flynn, M., T. Hoverd and D. Brazier (1989). Formaliser- An Interactive Support Tool for Z. Fourth Annual Z User Meeting, Oxford, Springer-Verlag.

FME. (2006). "FormalMethodsEurope (www.fmeurope.org - accessed in May 2006)."

Forcheri, P. and M. T. Molfino (1994). "Software Tools for the Learning of Programming: A proposal." Computers Education **23**(4): 269-278.

Fowler, M. (1998). UML Distilled; Applying the standard modelling language, Addison Wesley Longman Inc.

France, R., B, E. Grant and J. Bruel, M (2000). UMLtranZ: A UML-based rigorous object-oriented modelling technique. Fort Collins, Colorado, Colorado State University.

Fuchs, N. E. (1999). Specifications are (preferably) executable. High-Integrity System Specification and Design. J. P. Bowen and M. G. Hinchey. **12:** 583–607.

Gardner-Medwin, A. R. (1995). "Confidence assessment in the teaching of basic science." Association of Learning Technology Journal **3**: 80-85.

Gardner-Medwin, A. R. (2005). Enhancing Learning and Assessment Through Confidence-Based Learning Enhancing Teaching and Learning through Assessment, Hong Kong.

Gibson, P. and D. Mery (1998). Teaching Formal Methods: Lessons to learn. IWFM, Ireland.

Glaser, R. (1990). "The re-emergence of learning theory within instructional research." American Psychologist **45**(1): 29-39.

Goldsack, S., J and S. Kent , J, H (1996). Formal Methods and Object Technology, Springer-Verlag.

Gorard, S. (2001). Quantitative methods in educational research : the role of numbers made easy. London, Continuum.

Gosling, J. and H. McGilton (1996) "The Java Language Environment: Contents: A White Paper." http://java.sun.com/docs/white/langenv/ **Volume**, DOI:

Graeme, S. (2007). "Object-Z Web-Site Maintained by Graeme, Smith (accessed on 03, April 2007)."

Greer, J. E. and G. Koehn (1995). The peculiarities of plan recognition for intelligent tutoring systems. The Next Generation of Plan Recognition Systems: Challenges for and Insight from Related Areas of AI.

Greer, J. E. and G. McCalla (1994). Student modelling : the key to individualized knowledge-based instruction. Berlin ; New York, Springer-Verlag.

Hall, J., A (1990). Using Z as a specification calculus for object-oriented analysis. VDM'90- VDM and Z. B. Dines and C. Hoare, A, R, Springer Verlag.

Hall, J., A (1994). Specifying and Interpreting Class Hierarchies in Z. ZUM'94, Springer Verlag.

Hammond, J., A, R (1994). Producing Z specification from object-oriented analysis. Z User Meeting - ZUM'94.

Harel, D. and E. Gery (1997). "Executable Object Modeling with Statecharts." Computer, IEEE Press **30**(7.): 31-42 (cover feature).

Harel, D. and B. Rumpe (2004). "Meaningful modeling: what's the semantics of "semantics"?" Computer, IEEE Press **37**(10): 64- 72.

Hartley, J. R. and D. H. Sleeman (1973). "Towards Intelligent Teaching Systems." Int. J. of Man-Machine Studies.

Hatano, G. (1998). Comprehension Activity in Individuals and Groups. Advances in psychological sciences :Biological and cognitive aspects. M. Sabourin, F. Craik and M. Roberts. Hove, UK, Psychology Press. **2:** 399-417.

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

Hayes, I. J. and C. B. Jones (1999). Specifications are not (necessarily) executable. High-Integrity System Specification and Design. J. P. Bowen and M. G. Hinchey. **12:** 563-581.

Hedberg, J. G. and B. Harper (1995). Exploring the interactive multimedia information landscapes. ED-MEDIA'95, Austria.

Henrion, M., M. Pradhan, B. Del Favero, K. Huang, G. Provan and P. O'Rorke (1996). Why is diagnosis in belief networks insensitive to imprecision in probabilities? 12th International Conference on Uncertainty in Artificial Intelligence.

Hogan, K. and M. E. Pressley (1997). Scaffolding Student Learning: Instructional Approaches and Issues. The University of Albany, State University of New York, BROOKLINE.

Hohmann, L., M. Guzdial and E. Soloway (1992). SODA: A computer-aided design environment for the doing and learning of software design. Fourth International Conference on Computer Assisted Learning, Nova Scotia, Canada.

Holt, P., S. Dubs, M. Jones and J. Greer (1993). The State of Student Modelling. Student Modelling: Key to Individualized Instruction. J. Greer and G. McCalla, NATO Conference on Student Modelling.

Hopgood, A. (2000). Intelligent systems for engineers and scientists. Boca Raton, FL, CRC Press.

Hundhausen, C. D., S. A. Douglas and J. T. Stasko (2002). "Meta-Study of Algorithm Visualization Effectiveness." J. of Visual Languages & Computing **13**(3): pp. 259–290.

Jackson, D. and M. Vaziri (1999). Some shortcomings of OCL, the object constraint language of UML, MIT Laboratory for Computer Science.

Jameson, A. (1996). "Numerical Uncertainty Management in User and Student Modelling: An overview of Systems and Issues." User Modelling and User-Adapted Interaction **5**.

JDatastore (2006). Borland's Official Web site (http://www.borland.com/us/products/jdatastore/index.html)

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.

Jitgarun, K. (2004). Factors Affecting Students' Learning According to Constructionism Theory. Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2004. P. Kommers and G. Richards. Chesapeake, VA, AACE**:** pp. 2414-2419.

Johnson, W. L. and E. Soloway (1985). PROUST: An automated debugger for Pascal programs. BYTE. **10:** 179-190.

Jonassen, D. H. (1990). "Thinking Technology: Toward a constructivist view of instructional design." Education Technology **30**(9): 32-34.

Jonassen, D. H. (1995). Learning by Technology: Computers as Cognitive Tools. Handbook of Research on Educational Communications and Technology. H. Jonassen, D., Scholastic Publications.

Kanuka, H. and T. Anderson (1999). "Using Constructivism in Technology-Mediated Learning." Radical Pedagogy **1**(2).

Katz, S., A. Lesgold, G. Eggan and M. Gordin (1992). "Modelling the Student in Sherlock-II." Artificial Intelligence in Education **4**(3): 495-518.

Katz, S., A. Lesgold, G. Eggan and M. Gordin (1994). Modelling the Student in Sherlock-II. Student modelling : the key to individualized knowledge-based instruction. J. E. Greer and G. McCalla. Berlin ; New York, Springer-Verlag**:** 495-518.

Kay, J. (2000). "Stereotypes, student models and scrutability." Intelligent Tutoring Systems, Proceedings **1839**: 19-30.

Kay, J., A. Lum and D. Zapata-Rivera, Eds. (2005). Learner Modelling for Reflection, to Support Learner Control, Metacognition and Improved Communication; Proceedings of the Workshop (LEMORE) at the International Conference on Artificial Intelligence in Education 2005 (AIED'05). Amsterdam, Netherlands, IOS Press.

Keller, J. M. (1983). Motivational design of instruction. Instructional design theories and models: An overview of their current status. C. M. Reigeluth. Hillsdale, NJ:, Lawrence Erlbaum.

Kelly, G. A. (1955). The Psychology of Personal Constructs. NY, Norton.

Kemp, R. H., T. Stewart, I. P. W. Fung and B. Orban (2001). "Learning by Creating: Letting the Student Do the Work." Interactive Learning Environments.

Kemp, R. H., E. Todd and Y. Lu (2003). A Novel Approach on Teaching an Understanding of Programming. Artificial Intelligence in Education: Shaping the Future of Learning through Intelligence in Education. J. Kay. Sydney, IOS Press**:** 449-451.

Kiat, P. P. (2003). Decision-Theoretic Intelligent Tutoring Systems. Industrial & Systems Engineering. Singapore, National University of Singapore.

Kim, S. K. and D. Carrington (1999). "Formalizing the UML class diagram using Object-Z." UML1999, LNCS **1723**(Springer: Berlin): pp. 83-98.

Kim, S. K. and D. Carrington (2000a). "A Formal Mapping between UML Models and Object-Z Specifications." ZB2000, LNCS **1878**(Springer: Berlin): pp. 2-21.

Kim, S. K. and D. Carrington (2000b). UML Metamodel Formalization with Object-Z: the State Machine Package, Technical Report 00-29, SVRC. The University of Queensland, Australia.

Kim, S. K. and D. Carrington (2002). A Formal Metamodeling Approach to transformation between the UML State Machine and Object-Z. In International Conference on Formal Engineering Methods (ICFEM 2002), Lecture Notes in Computer Science. Springer-Verlag.

Kim, S. K. and D. Carrington (2005). An MDA Approach towards Integrating Formal and Informal Modeling Languages. Formal Methods 2005, LNCS 2495 . Springer-Verlag, UK.

Koschmann, T. D. (1996). CSCL, theory and practice of an emerging paradigm. Mahwah, N.J., L. Erlbaum Associates.

Kulhavy, R. W. (1977). "Feedback in Written Instruction." Review of Educational Research **47**(1): 211-232.

Kulhavy, R. W., A. Stock, W,, E. Hancock, T, K. Swindell, L and P. Hammrich, L (1990). "Written Feedback: Response Certitude and Durability." Contemporary Educational Psychology **15**: 319-332.

Kulhavy, R. W. and A. W. Stock (1989). "Feedback in Written Instruction: The place of certitude estimation." Educational Psychology Review **1**(1): 279-308.

Kulhavy, R. W., M. T. White, B. W. Topp, A. L. Chan and J. Adams (1985). "Feedback complexity and corrective efficiency." Contemporary Educational Psychology **10**: 285-291.

Kulik, J. A., Chen-Lin and C. Kulik (1988). "Timing of Feedback and Verbal Learning." Review of Educational Research **58**(1): 79-97.

Lane, H. C., and VanLehn, K. (2003). Coached program planning: Dialogue-based support for novice program design. In Proceedings of the. Thirty-Fourth Technical Symposium on Computer Science Education (SIGCSE), Reno, NV.

Lano, K., C (1991). Z++, An Object-orientated Extension to Z. Z User Workshop. J. Nicholls, E, Springer-Verlag.

Lano, K. and S. Goldsack, J (1996). Integrated Formal and Object-oriented Methods: The VDM++ Approach. In proceedings of Methods Integration Workshop, Supported by BCS FACS (Leeds Metropolitan University).

Laurillard, D. (1996). "Changing University." ITForum Retrieved 05/07/2005, 2005.

Laurillard, D. (2002). Rethinking University Teaching: A Conversational Framework for the Effective Use of Learning Technologies. London:, Routledge.

Laurillard, D. (2006). E-Learning in Higher Education. Changing higher education: The development of learning and teaching (in press). P. Aswin. London, Routledge**:** 71-86.

Lave, J. and E. Wenger (1990). Situated Learning: Legitimate Peripheral Participation. Cambridge, UK, Cambridge University Press.

Lehtinen, E., K. Hakkarainen, L. Lipponen, M. Rahikainen and H. Muukkonen (1999). "Computer-supported collaborative learning: A review of research and development." The J.H.G.I. Giesbers Reports on Education, University of Nijmegen **10**.

Lipponen, L., K. Hakkarainen and S. Paavola (2004). Practices and orientations of CSCL. What We Know About CSCL and Implementing It in Higher Education. J. Strijbos, P. A. Kirschner, R. L. Martens and P. Dillenbourg. Norwell, Kluwer Academic Publishers. **3:** 31-50.

Macromedia (2001). Using Authorware, Macromedia, Inc.., 600 Townsend St. San Francisco.

Martin, J. and K. VanLehn (1993). OLAE: Progress Towards a Multi-activity, Bayesean Student Modeler., AACE.

Mason, J. B. and R. Bruning. (1999). "Providing Feedback in Computer-based Instruction: What the research tells Us. ." Retrieved Feb 17, 2004, from http://www.ccinul.edu/Edit/MB/MasonBruning.html.

Mayo, M. (2001). Bayesian Student Modelling and Decision-Theoretic Selection of Tutorial Actions in ITS (Chapter 1, Pg 6). Computer Science. Canterbury, University of Canterbury, New Zealand**:** 124-153.

Mayo, M. and A. Mitrovic (2001). "Optimising ITS behaviour with Bayesian networks and decision theory." International Journal of Artificial Intelligence in Education (IJAIED) **12**(2): 124-153.

McArthur, D., M. Lewis and M. Bishay (1999). The roles of AI in education: Current progress and future prospects, RAND cooperation, Santa Monica, CANADA.

McCalla, G. I. and J. E. Greer (1994). Granularity-- based reasoning and belief revision in student models. Student Modelling: The Key to Individualized Knowledge--Based Instruction,. J. E. Greer and G. I. McCalla. Springer--Verlag, Berlin.**:** pages 39--62.

McComb, T. and G. Smith (2003). Animation of Object-Z Specifications Using a Z Animator. International Conference on Software Engineering and Formal Methods (SEFM 2003), Brisbane, Australia.

Mellor, S., J and M. Balcer, J (2002). Executable UML:a foundation for model-driven architecture / Stephen J. Mellor, Marc J. Balcer. Boston, Addison-Wesley.

Mergel, B. (1998). "Instructional Design and Learning Theory (accessed in May, 2006); http://www.usask.ca/education/coursework/802papers/mergel/brenda.htm."

Merrill, M. D. (1980). "Learner control in computer based learning." Computers and Education **4**: 77-95.

Merrill, M. D. (1983). Component Display Theory. Instructional Design Theories and Models. C. Reigeluth. Hillsdale, NJ, Erlbaum Associates.

Miao, H., L. Lui and L. Li (2002). Formalizing UML Models with Object-Z. International Conference on Formal Engineering Methods (ICFEM 2002), Lecture Notes in Computer Science. Springer-Verlag.

Mikusiak, L., J. Hasaralejeko and D. Koronthaly (1995). Z Browser - Tool for Visualization of Z Specifications. ZUM'95 - 9th International Conference of Z Users,, Springer-Verlag.

Miller, G. A. (1956). "The magical number seven, plus or minus two: Some limits on our capacity for processing information." Psychological Review **63**(81-97).

Miller, M. L. (1978). "A structured planning and debugging environment for elementary programming." International Journal of Man-Machine Studies **11**: 79-95.

Mislevy, R. J. and D. H. Gitomer (1996). "The Role of Probability-Based Inference in an Intelligent Tutoring System." User Modelling and User-Adapted Interaction **5**: pp. 253-282.

Mitrovic, A. (2005). Constraint-based tutors: a success story. - Brief biography (key note speech). AIED' 05.

Mitrovic, A. and B. Martin (2002). Evaluating the Effects of Open Student Models on Learning. Adaptive Hypermedia and Adaptive Web-Based Systems: Second International Conference, AH 2002, Malaga, Spain.

Moreno, A., N. Myller, E. Sutinen and M. Ben-Ari (2004). Visualizing Programs with Jeliot 3. International Working Conference on Advanced Visual Interfaces AVI 2004, Gallipoli, Italy.

Morrey, I., J. Siddiqi, I, A, R. Hibberd and G. Buckberry (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.

Mory, H. E. (1996). Feedback Research. Handbook of Research on Educational Communications and Technology. D. Jonassen. New York, Printice-Hill.

Munro, A., M. Johnson, C, Q. Pizzini, A, D. Surmon, S, D. Towne, M and J. Wogulis, L (1997). "Authoring simulation centered tutors with RIDES." International Journal of Artificial Intelligence in Education **8**(3-4): 284-316.

Murray, R. C. and K. VanLehn, Eds. (2000). DT Tutor: A decision-theoretic, dynamic approach for optimal selection of tutorial actions. ,. Intelligent Tutoring Systems, Fifth International Conference, ITS 2000. Montreal, Canada., New York: Springer.

Murray, T. (1999). "Authoring Intelligent Tutoring Systems: An analysis of the state of the art." International Journal of Artificial Intelligence in Education **10**: 98-129.

Murray, T., S. Blessing and S. E. Ainsworth (2003). Tools for Advanced Technology Learning Environments. Hillsdale, NJ, Erlbaum.

Ohlsson, S. (1994). Constraint--based Student Modeling. Student Modeling: the Key to Individualized Knowledge--based Instruction. J. E. Greer and G. I. McCalla, Springer-Verlag,. **125:** 167-189.

OMG (2006). "UML Standards v2.0 (UML Semantics) - accessed in June 2006." http://www.uml.org/#UML2.0.

Ormrod, J. E. (2003). Educational Psychology: Developing Learners, Fourth Edition., Prentice Hall.

Otsuki, S. (1993). Intelligent Environment for Discovery Learning. AI in Education, Charlottesville, VA, AACE.

Overgaard, G. (2000). Using the Boom Framework for Formal Specification of the UML. Defining Precise Semantics for UML, ECOOP'2000 workshop.

Paiva, A., J. Self and R. Hartley (1995). Externalising learner models. World Conference on Artificial Intelligence in Education, Washington, DC.

Palumbo, D. B. (1990). "Programming language/ problem solving research: A review of relevant issues." Review of Educational Research **60**(1): 65-89.

Papert, S. (1972). "Teaching Children Thinking." Programmed Learning and Educational Technology **9**(5): 245-255.

Papert, S. (1979). Final report of the Brookline LOGO Project. Cambridge, Mass., Massachusetts Institute of Technology Artificial Intelligence Laboratory.

Papert, S. (1991). Situated Constructionism. Constructionist Theory. S. Papert and I. Harel. Norwood, NJ, Ablex Publishing Corporation.

Pearl, J. (1988). Probabilistic reasoning in intelligent systems : networks of plausible inference. San Mateo, Calif., Morgan Kaufmann Publishers.

Piaget, J. (1968). Six Psychological Studies. New York, Vintage Books.

Pintrich, R., P and H. Schunk, D (2002). Motivation in Education (Second Edition). NJ, OHIO, Merrill Printice Hall.

Polack, F. (1992). "Integrating Formal Methods and System Analysis Using ER Diagrams." Software Engineering Journal **7**(5): 363-371.

Polack, F. (2000). SAZ: SSADM Version 4 and Z. Software Specification Methods: An Overview Using a Case Study. M. Frappier and H. Habrias, Springer.

Psotka, J., L. D. Massey and S. A. Mutter (1988). Intelligent tutoring systems : lessons learned. Hillsdale, NJ, L. Erlbaum Associates.

pUML (2006). "Precise UML Group (http://www.cs.york.ac.uk/puml/ or www.sosym.org)." accessed in June 2006.

Quillian, M. R. (1968). Semantic Memory. Semantic Information Processing. M. Minsky. Cambridge, MIT Press.

Ragnemalm, E. L. (1995). Student diagnosis in practice : bridging a gap. Linköping, Sweden, Linköping University Dept. of Computer and Information Science.

Ramadhan, H. (1992a). Intelligent vs unintelligent programming systems for novices. Sixteen Annual International Computer Software and Applications, IEEE, Chicago, Illinois.

Ramadhan, H. A. (1992b). Intelligent systems for discovery programming: Ph.D. Thesis. Brighton [East Sussex], University of Sussex School of Cognitive and Computing Sciences**:** xii, 188.

Ramasundaram, V., S. Grunwald, A. Mangeot, N. B. Comerford and C. M. Bliss (2005). "Development of an environmental virtual field laboratory." Computers & Education **45**(1): 21-34.

Ravid, R. (2000). Practical Statistics for Educators. Oxford, University Press of America.

Reed, S., K (2003). Cognition: Theory nad Applications (Sixth Edition). Belmont, CA, Wadsworth.

Reigeluth, C. M. (1999). The elaboration theory: guidance for scope and sequence decisions. Instructional Design Theories and Models: A New Paradigm of Instructional Theory. C. M. Reilgeuth. NJ, Lawrence Erlbaum Associates. **2:** 425-453.

Reye, J. (1995). A goal-centered architecture for intelligent tutoring systems. World Conference on Artificial Intelligence in Education- AIED'95, Washington, D. C., USA.

Reye, J. (2004). "Student Modelling based on Belief Networks." International Journal of Artificial Intelligence in Education **14**(63-96).

Romero, B., B. du Boulay, R. Lutz and R. Cox (2003). The effects of graphical and textual visualizations in multi-representational debugging environments. IEEE Symposia on Human Centric Computing Languages Environments.

Russell, D. (1988). IDE: The interpreter. Intelligent Tutoring Systems, Lessons Learned. J. Psotka, L. Massey and S. A. Mutter. Hillsdale, NJ, Lawrence Erlbaum.

Russell, S. J. and P. Norvig (2003). Artificial intelligence : a modern approach. Upper Saddle River, N.J., Prentice Hall/Pearson Education.

Ryan, R. M. and N. Dean, C. (2000). "Self-Determination Theory and the Facilitation of Intrinsic Motivation, Social Development, and Well-Being." American Psychologist **33**(1): 68-78.

Saaltink, M. (1997). Z-Eves System: The Formal Specification Notation, Springer.

Salomon, G. (1993). On the nature of Pedagogic Computer Tools: The case of the *Writing Partner*. Computers as Cognitive Tools.

Self, J. (1988). Artificial intelligence and human learning : intelligent computer-aided instruction. London ; New York, Chapman and Hall.

Self, J. (1990). Bypassing the intractable problem of student modelling. Intelligent tutoring systems: at the crossroads of artificial intelligence and education. C. Frasson and G. Gauthier. Norwood, New Jersey, Ablex Publishing**:** 107-123.

Self, J. (1999). "The defining characteristics of intelligent tutoring systems research: ITSs care, precisely." International Journal of Artificial Intelligence in Education **10**: 350-364.

Shapiro, E., Y. (1982). Algorithimatic Program Debugging. Cambridge, MA, MIT press.

Shlaer, S. and S. J. Mellor (1988). Object-Oriented Systems Analysis: Modeling the World in Data, Yourdon Press

Shneiderman, B. (1977). "Teaching programming: A spiral approach to syntax and semantics." Computers and Education **1**: 193-197.

Shneiderman, B. and C. Plaisant (2004). Designing the User Interface: Strategies for Effective Human-Computer Interaction: Fourth Edition Reading, MA, Addison-Wesley Publ. Co. .

Shroff, M. and R. France, B (1997). Towards a Formalization of UML Class Structures in Z. COMPSAC'97.

Shute, V. J. and Air Force Human Resources Laboratory. (1990a). Individual differences in learning from an intelligent discovery world

Smithtown. Brooks Air Force Base, Tex., Air Force Systems Command Air Force Human Resources Laboratory.

Shute, V. J. and R. Glaser (1990b). "A large-scale evaluation of an intelligent discovery world: Smithtown." Interactive Learning Environments **1**: pp. 51-76.

Shute, V. J. and J. Psotka (1995). Intelligent Tutoring Systems: Past, Present, and Future. Handbook of Research on Educational Communications and Technology. H. Jonassen, D., Scholastic Publications.

Simkin, G., M and L. Kuechler, W (2005). "Multiple-Choice Tests and Student Understanding: What is the Connection?" Decision Sciences Journal of Innovative Education, USA. **3**(1).

Skinner, B. F. (1954). "The science of learning and the art of teaching." Harvard Educational Review **24**(2): 86-97.

Skinner, B. F. (1958). "Teaching Machines." Science **128**.

Sleeman, D. (1987). PIXIE: A shell for developing intelligent tutoring systems. Artificial Intelligence and Education. R. Lawler and M. Yazdani. Norwood, NJ, Ablex. **1**.

Smith, G. (2000). The Object-Z Specification Language, Kluwer Academic Publishers.

Soloway, E., K. Ehrlich, J. Bonar and J. Greenspan (1982). What do novices know about programming. Directions in Human-computer Interaction. A. Badre and B. Shneiderman. Norwood; NJ, Ablex publishing.

Sommerville, I. (2001). Software Engineering. Harlow, England, Addison-Wesley.

Spiro, R. J., P. J. Feltovich, M. J. Jacobson and R. L. Coulson (1992). Cognitive flexibility, Constructivism and Hypertext: Random access instruction for advanced knowledge acquisition in ill-structured domains. Constructivism and the Technology of Instruction. T. Duffy and D. Jonassen. Hillsdale, NJ, Erlbaum.

Steggles, P. and J. Hulance (1994). Z Tools Survey, Imperial Software Technology Ltd

Formal Systems (Europe) Ltd.

Stepney, S., R. Barden and D. Cooper (1992). Object Orientation in Z. London, Springer-Verlag.
Stern, M., J. Beck and B. P. Woolf (1999). Naive Bayes Classifiers for User Modeling. User Modelling- UM'99.
Stern, M. and B. P. Woolf (1998). "Curriculum Sequencing in a Web-Based Tutor." Lecture Notes in Computer Science **1452**: 574-599.
Sun, J., S. Dong J, J. Liu and H. Wang (2001). Z family on the web with their UML photos. TRA1-01. Singapore, School of Computing, National University of Singapore.
Suraweera, P. and A. Mitrovic (2004). "An Intelligent Tutoring System for Entity Relationship Modelling
" International Journal of Artificial Intelligence in Education **14**: 375-417.
Sweller, J. (1988). "Cognitive load during problem solving: Effects on learning,." Cognitive Science, . **12**: 257-285.
Sweller, J. (1994). "Cognitive load theory: learning difficulty and instructional design." Learning and Instruction **4**: 295-312.
Taheri, M. S. (2003). "Trends in Fuzzy Statistics." Australian Journal of Statistics **32**(3): 239-257.
Tekinerdogan, B. and H. P. M. Krammer (1995). Design of a modular composable tutoring shell for imperative programming languages. International Conference on Computer in Education. ICCE 95., Singapore, Assoc. Advancement of Comput. Educ. Charlottesville, VA, USA.
Thorndike, E. (1913). Educational Psychology: The Psychology of Learning. New York, Teachers College Press.
Tremblay, G. (2000). "Formal Methods: Mathematics, Computer Science or Software Engineering." IEEE Transactions on Education **43**(4).
Turnbull, A., R. Turnbull, M. Shank and D. Leal (1999). Exceptional lives : special education in today's schools. Upper Saddle River, N.J., Printice- Hall, Inc:.
Ueno, H. (1994). "Integrated intelligent programming environment for learning programming." IEICE Transactions on Information and Systems **E77-D**(1): 68-79.
Uhr, L. (1969). Teaching machine programs that generate problems as a function of interaction with students. 24th National ACM Conference, New York, ACM.
Utting, M. and W. Shaochun (2002). Object-Orientation in Standard Z. Hamilton, New Zealand, Department of Computer Science
University of Waikato.
VanLehn, K. (1982). "Bugs are not enough: Empirical studies of bugs, impasses, and repairs in procedural skills." Journal of Mathematical Behaviour **3**(2): 3-72.
VanLehn, K., C. Lynch, K. Schulze, J. Shapiro, R. Shelby, L. Taylor, D. Treacy, A. Weinstein and M. Wintesgill (2005). "The Andes Physics Tutoring System: Five Years of Evaluations." International Journal of Artificial Intelligence in Education **15**(3): 1-47.
Villano, M. (1992). Probabilistic Student Models: Bayesian Belief Networks and Knowledge Space Theory. Proceedings of the Second International Conference on Intelligent Tutoring Systems, Springer-Verlag.
Vygotsky, L. S. (1978). Mind in Society. Cambridge, MA, Harvard University Press.
Weaver, P. L. (1993). Practical SSADM Pitman, London.
Weber, G. (1993). Analogies in an intelligent programming environment for learning LISP. Cognitive Models and Intelligent Environments for Learning Programming. E. Lemut, B. du Boulay and G. Dettori. Berlin: Springer-Verlag**:** , pp. 210-219,.
Weber, G. and M. Specht (1997). User modeling and adaptive navigation support in WWW-based tutoring systems. Sixth International Conference on User Modeling, UM97, Sardinia, Italy.
Wong, W. K., T. W. Chan, C. Y. Chou, J. S. Heh and S. H. Tung (2003). "Reciprocal tutoring using cognitive tools." Journal of Computer Assisted Learning **19**(4): pp. 416-428.
Wood, D. J., J. S. Bruner and G. Ross (1976). "The role of tutoring in problem solving." Journal of Child Psychology and Psychiatry **17**(2): 89-100.
WordNet. (2005). "A lexical database for the English language (http://wordnet.princeton.edu/) Accessed in June 2006." Retrieved 01/07/05, 2005.
WorldWeb. (2005). "wordwebonline.com/en/MENTALSTATE." Retrieved 01/07/05, 2005.

Xu, S. and Y. S. Chee (2003). "Transformation-Based Diagnosis of Student Programs for Programming Tutoring Systems." IEEE Trans. Softw. Eng. **29**(4): 0098-5589.

Yap, C. N. (1999). "Visual-Z: a Methodology and Environment for Developing Visual Formal Z Specifications." Ph.D. thesis.

Zadeh, L. (1973). "Outline of a new approach to the analysis of complex system and design processes." IEE Trans. Syst. Man Cybernet. **SMC-3**: pp. 28-44.

Zadeh, L. (1994). "Fuzzy logic, Neural networks, and soft computing." Communications of ACM **37**(3): 77-84.

Zapata-Rivera, J.-D. and J. Greer (2004). "Inspectable Bayesian student modelling servers in multi-agent tutoring systems." J Int. J. Hum.-Comput. Stud. **61**(4): 535-563.

Appendix A: Survey on Z and Formalizer

A.1 Questionnaire

The questionnaire contains 18 statements. Except the first two, all the other statements use five-point Likert-type scale. The first statement is about the participants' levels of knowledge in Mathematics, and the second is about their levels of experience in software development. The next five statements are about the Z notation, and the rest are about the Formalizer. The last two statements have four parts each.

Date: May 30, 2002

Instruction: This questionnaire has eighteen statements in three sections. Please mark one option only.

Section 1: General

1. My knowledge level of mathematics (relevant to formal methods)

☐ Naive ☐ Moderat ☐ Expert

2. My work experience in software development field

☐ None ☐ Less than two years ☐ More than two vears

Section 2: The Formal Method Z

3. Z helps me to simplify complex system analysis problems.

☐ Strongly agree ☐ Agree ☐ Neutral ☐ Disagree ☐ Strongly disagree

4. Z specification may be verified for correctness.

☐ Strongly agree ☐ Agree ☐ Neutral ☐ Disagree ☐ Strongly disagree

5. Z helps me to understand the importance of abstraction in software problem analysis.

Strongly agree	Agree	Neutral	Disagree	Strongly disagree
☐	☐	☐	☐	☐

6. Z is a difficult subject to learn.

Strongly agree	Agree	Neutral	Disagree	Strongly disagree
☐	☐	☐	☐	☐

7. I will use Z for specification, if I ever need to specify a critical system.

Strongly agree	Agree	Neutral	Disagree	Strongly disagree
☐	☐	☐	☐	☐

Section 3: The Formalizer Tool

8. Formalizer helped me to learn Z.

Strongly agree	Agree	Neutral	Disagree	Strongly disagree
☐	☐	☐	☐	☐

9. Formalizer is difficult to learn.

Strongly agree	Agree	Neutral	Disagree	Strongly disagree
☐	☐	☐	☐	☐

10. Formalizer motivated me to learn Z.

Strongly agree	Agree	Neutral	Disagree	Strongly disagree
☐	☐	☐	☐	☐

11. Formalizer is annoying to use.

☐	☐	☐	☐	☐
Strongly agree	Agree	Neutral	Disagree	Strongly disagree

12. Formalizer helped me to understand the complexity of Z specification construction.

☐	☐	☐	☐	☐
Strongly agree	Agree	Neutral	Disagree	Strongly disagree

13. I hate learning Z because Formalizer is very hard to use.

☐	☐	☐	☐	☐
Strongly agree	Agree	Neutral	Disagree	Strongly disagree

14. Formalizer helped me to grasp the general idea of formal methods.

☐	☐	☐	☐	☐
Strongly agree	Agree	Neutral	Disagree	Strongly disagree

15. The error messages in Formalizer are not useful.

☐	☐	☐	☐	☐
Strongly agree	Agree	Neutral	Disagree	Strongly disagree

16. Formalizer helped me to understand the usefulness of Z in software development.

☐	☐	☐	☐	☐
Strongly agree	Agree	Neutral	Disagree	Strongly disagree

17. The following features of Formalizer are particularly helpful to use.

a. Symbols can be easily manipulated.

☐	☐	☐	☐	☐
Strongly agree	Agree	Neutral	Disagree	Strongly disagree

b. Structures are drawn automatically.

Strongly agree	Agree	Neutral	Disagree	Strongly disagree

c. Syntax directed editor gives on-the-ply alerts.

Strongly agree	Agree	Neutral	Disagree	Strongly disagree

d. Types and Scopes of objects may be checked easily.

Strongly agree	Agree	Neutral	Disagree	Strongly disagree

18. The following features, if added to Formalizer, would make it more enjoyable (not only to use it for Z specification construction but also to learn Z notation).

a. An 'on-request/context-sensitive' help system (as in the systems like MSWORD).

Strongly agree	Agree	Neutral	Disagree	Strongly disagree

b. An 'context-sensitive coach' (see the attached page)- in order to help me with not only syntax but also logical errors.

Strongly agree	Agree	Neutral	Disagree	Strongly disagree

c. An 'on-line coach' to help me to learn Z notation while using Formalizer

Strongly agree	Agree	Neutral	Disagree	Strongly disagree

d. A facility to validate specification by 'running' them.

us Strongly agree	Agree	Neutral	Disagree	Strongly disagree

Any other comments on Z or Formalizer:

_____

_____

_____

_____

_____

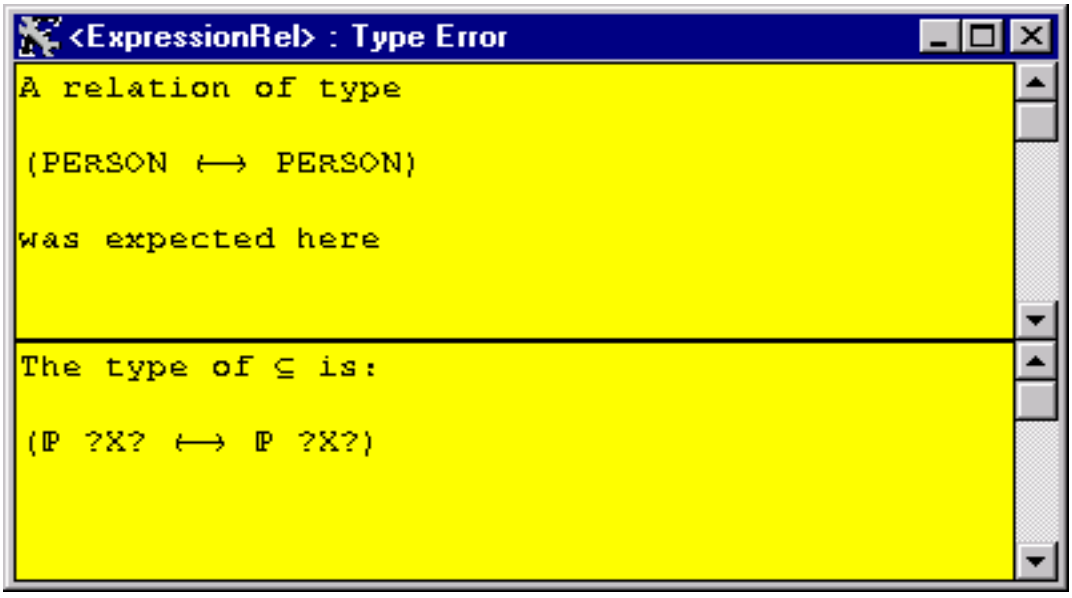
Help in Formalizer

This is the help screen Formalizer gives for the following specification.

<PERSON>

users, login : PERSON

users ⊆ login



Proposed Context-Sensitive Help

A help screen that a context-sensitive coach could generate in the same situation

A TYPE ERROR IDENTIFIED!

Error must be in one of the following two expressions.

users, logIn : PERSON

LogIN ⊆ users

DISCUSSION

The variables 'users' and 'logIn' are declared as PERSON type. Where PERSON is a basic type and is a single entity. It is not a collection of entities.

Since you have used a set notation (and also you have used a plural word 'users'), The system assumes that you want both variables 'users' and 'logIn' are to be declared as set of persons.
Instead, you declared,

users, loggedIn : PERSON

If you want both are to be set of PERSON, how can you declare them?

CHECK HELP

OTHERWISE,
If you insist that both are to be of base type PERSON, check the operation logIn ⊆ users (this is not possible in sets)

One suggestion is LogIn = users

SKIP BACK NEXT HELP ENTER

A.2 Responses

The questionnaire was given to 32 third year software engineering students and 29 of them had returned the completed questionnaire. The percentages given in the last two rows refer to the responses in the first two (agree) or last two points (disagree) in the scale.

Question Number		No. of Responses for each Lickert Level					SD	Avg	Agree (%)	Disagree (%)
		one	two	three	four	five				
Sec1	1	1	1	27	-	-				
	2	15	8	6	-	-				
Sec2	3	1	7	17	4	0	0.7	2.83	27.6	13.79
	4	1	19	8	1	0	0.6	2.31	69.0	3.45
	5	0	12	13	3	1	0.8	2.76	41.4	13.79
	6	5	7	10	6	1	1.1	2.69	41.4	24.14
	7	5	7	11	4	2	1.1	2.69	41.4	20.69
Sec3	8	0	8	8	8	5	1.1	3.34	28.0	44.83
	9	5	12	10	1	1	0.9	2.34	58.6	6.90
	10	0	1	9	12	7	0.8	3.86	3.5	65.52
	11	13	14	2	0	0	0.6	1.62	93.1	0.00
	12	1	11	12	4	1	0.9	2.76	41.4	17.24
	13	6	9	10	4	0	1	2.41	51.7	13.79
	14	0	13	6	7	3	1.1	3.00	44.8	34.48
	15	9	11	5	4	0	1	2.14	69.0	13.79
	16	0	3	14	8	4	0.9	3.45	10.3	41.38
	17a	0	7	5	12	4	1	3.46	24.1	55.17
	17b	0	17	7	4	0	0.7	2.54	58.6	13.79
	17c	0	8	13	6	0	0.7	2.93	27.6	20.69
	17d	1	10	10	6	1	0.9	2.86	38.0	24.14
	18a	8	18	3	0	0	0.6	1.83	89.67	0.00
	18b	9	14	6	0	0	0.7	1.90	79.3	0.00
	18c	12	15	1	1	0	0.7	1.69	93.1	3.45
	18d	7	19	3	0	0	0.6	1.86	89.6	0.00

Appendix B: Prototype Screen Shots

This appendix includes a sequence of screen shots as a student interacts with the system. The screen shots serves two purposes; firstly, it gives an idea of the organization of the lesson material on 'Visibility List in Object-Z', secondly, it introduces the interface of the prototype.

In order to use the system, a learner needs to login (Figure B.1), or if he/she is a new they have to register first (Figure B.2). After that, the content page will be displayed (Figure B.3). Students may start with the first lesson or may select any other desired lesson. They may also inspect the initial state of the learner model (Figures B.4a and B.4b).

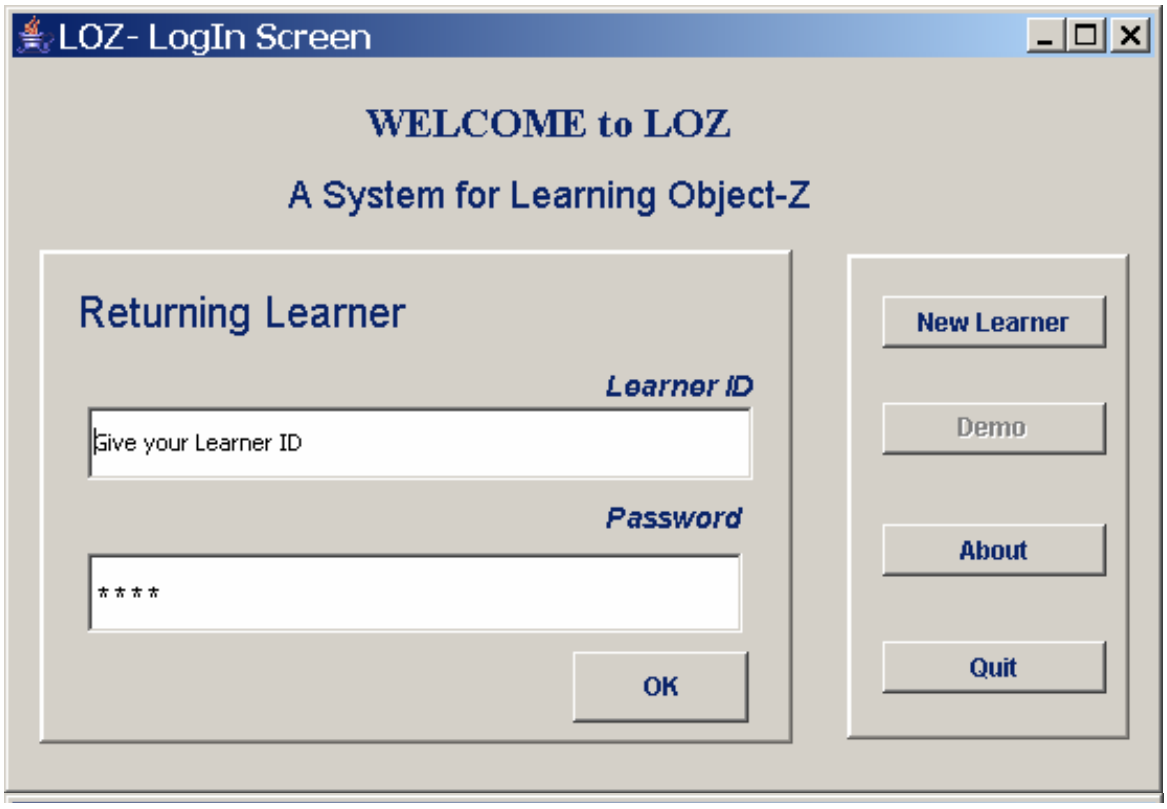


Figure B.1 Log-In Screen

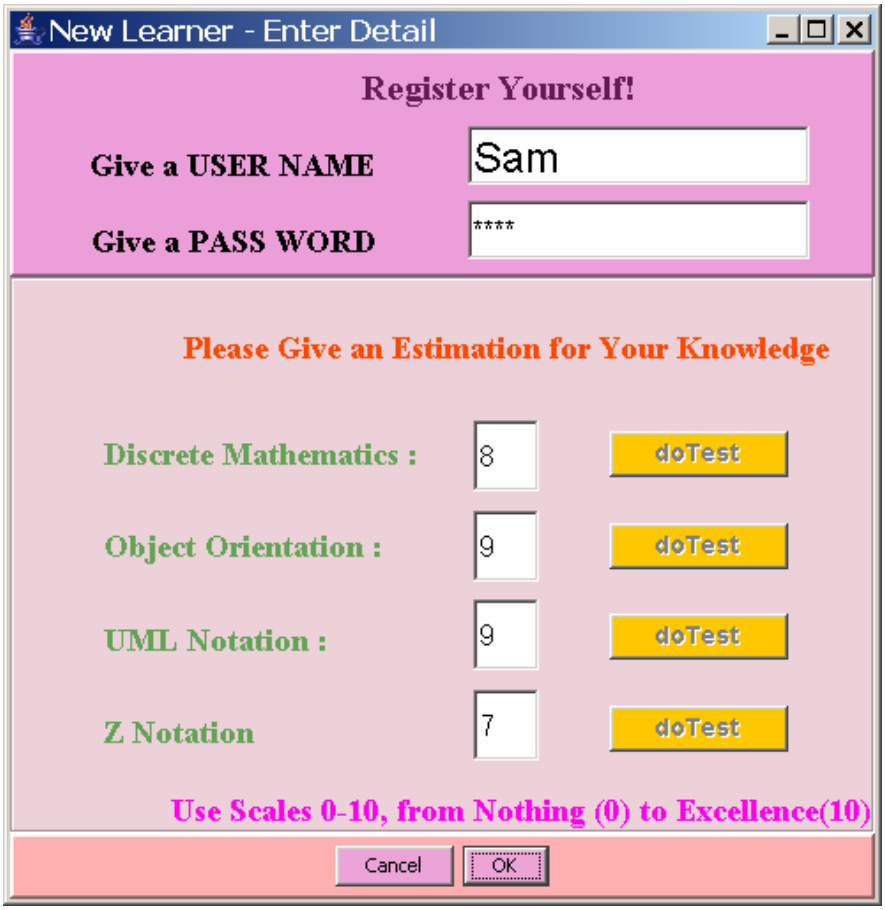


Figure B.2 New Learner Screen

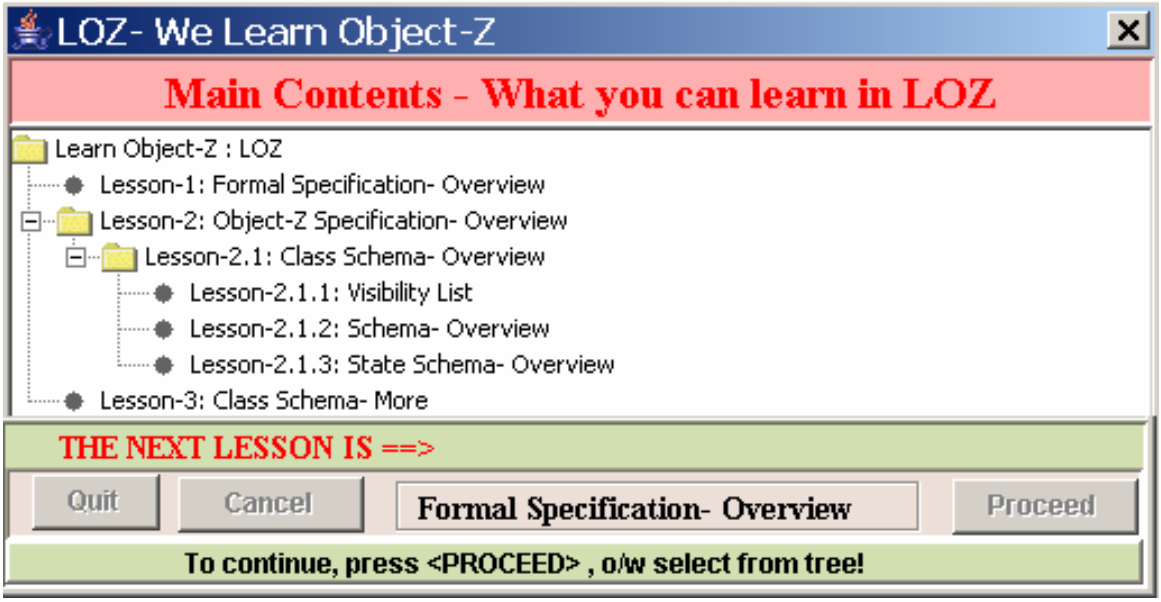
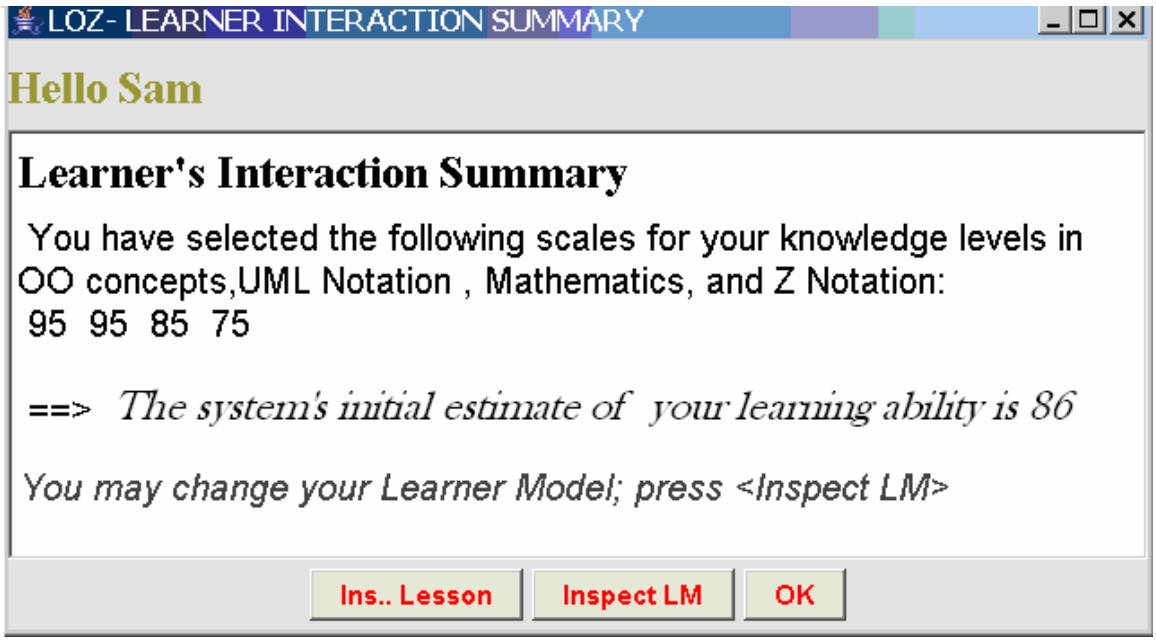
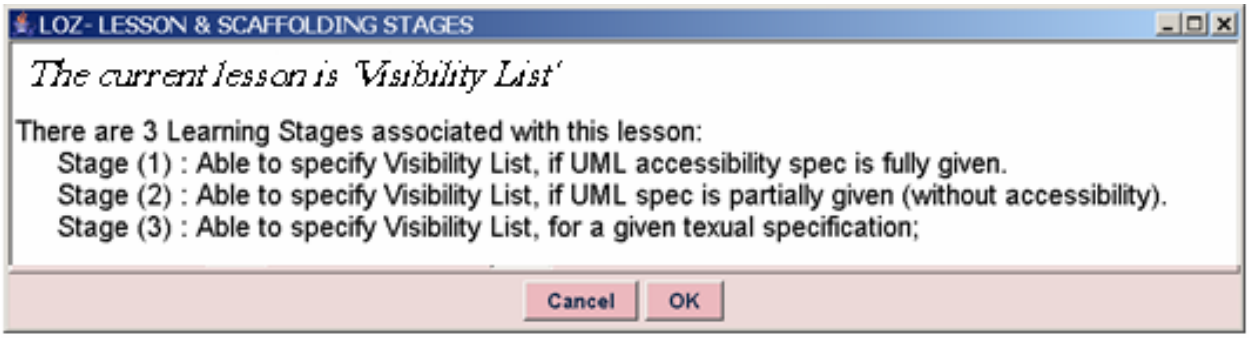


Figure B.3 Content Page





Figures B.4a and B.4b Inspecting Initial Learner Model

Initially, the system displays revision material relevant to Visibility List lesson (here it is UML accessibility options (Figure B.5)). There are two sub-lessons on Visibility List. System displays the first lesson next (Figure B.6). There are three scaffolding levels at the building stage. The learner may check this information (B.4b). There are more than one MC tests associated with each level. At the first level, questions are easy as they usually include hints using UML notations. At the later levels, however, questions are comparatively hard.

At the first level, the learner may answer correctly or incorrectly for the first MC test (Figure B.7). For correct answer, the system gives different feedback for different learners based on their initial estimates (Figures B.8 and B.9). Similarly, feedback may vary for incorrect answers also. Based on the system's estimates, the learner may be asked to try the same question again (Figure B.10).or try a similar question at the same level The learner may check the relevant Java code (Figure B.11). Finally, the system updates the learner model based on the new evidences (Figure B.12)

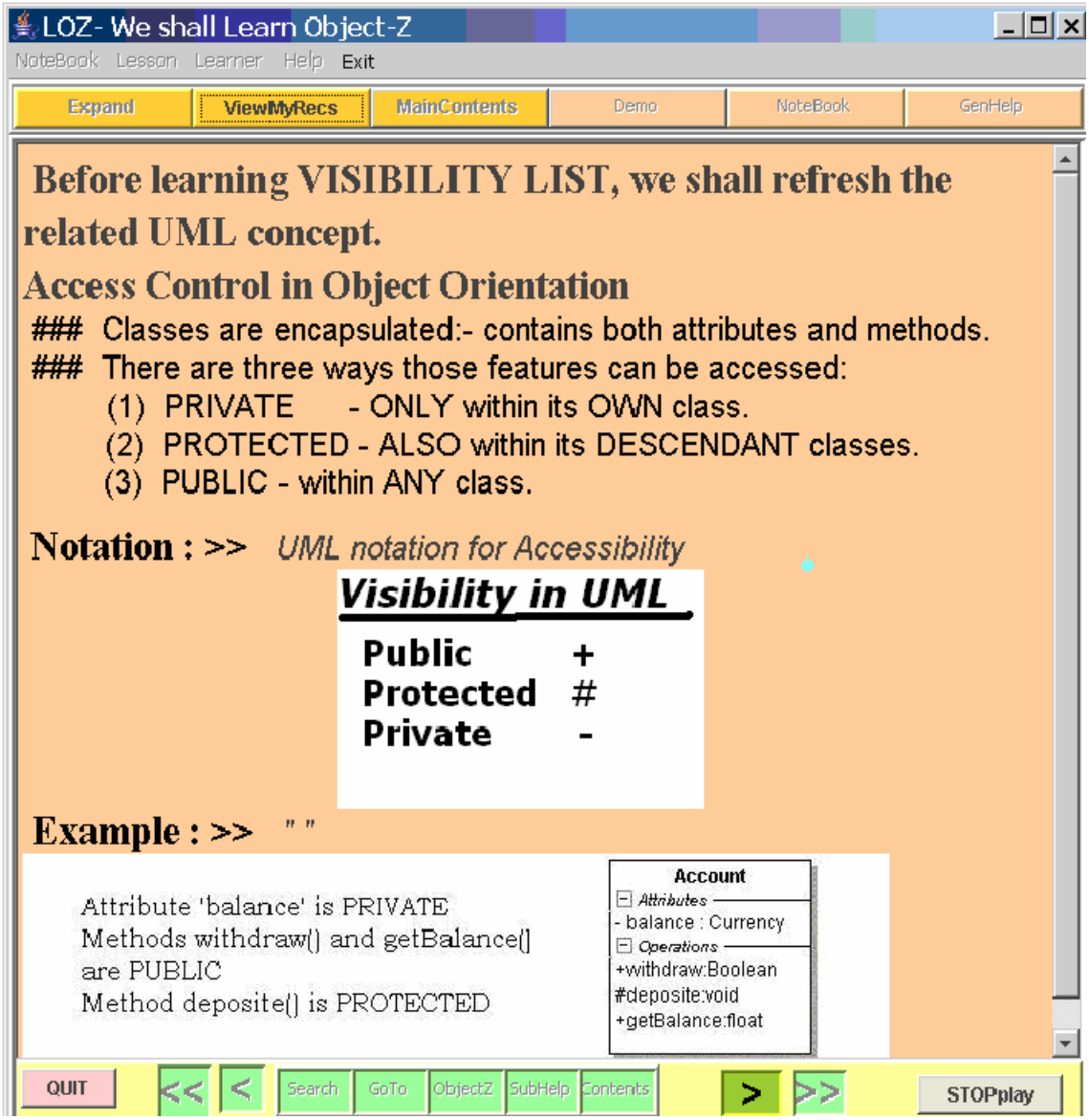


Figure B.5 Re-Conditioning Page

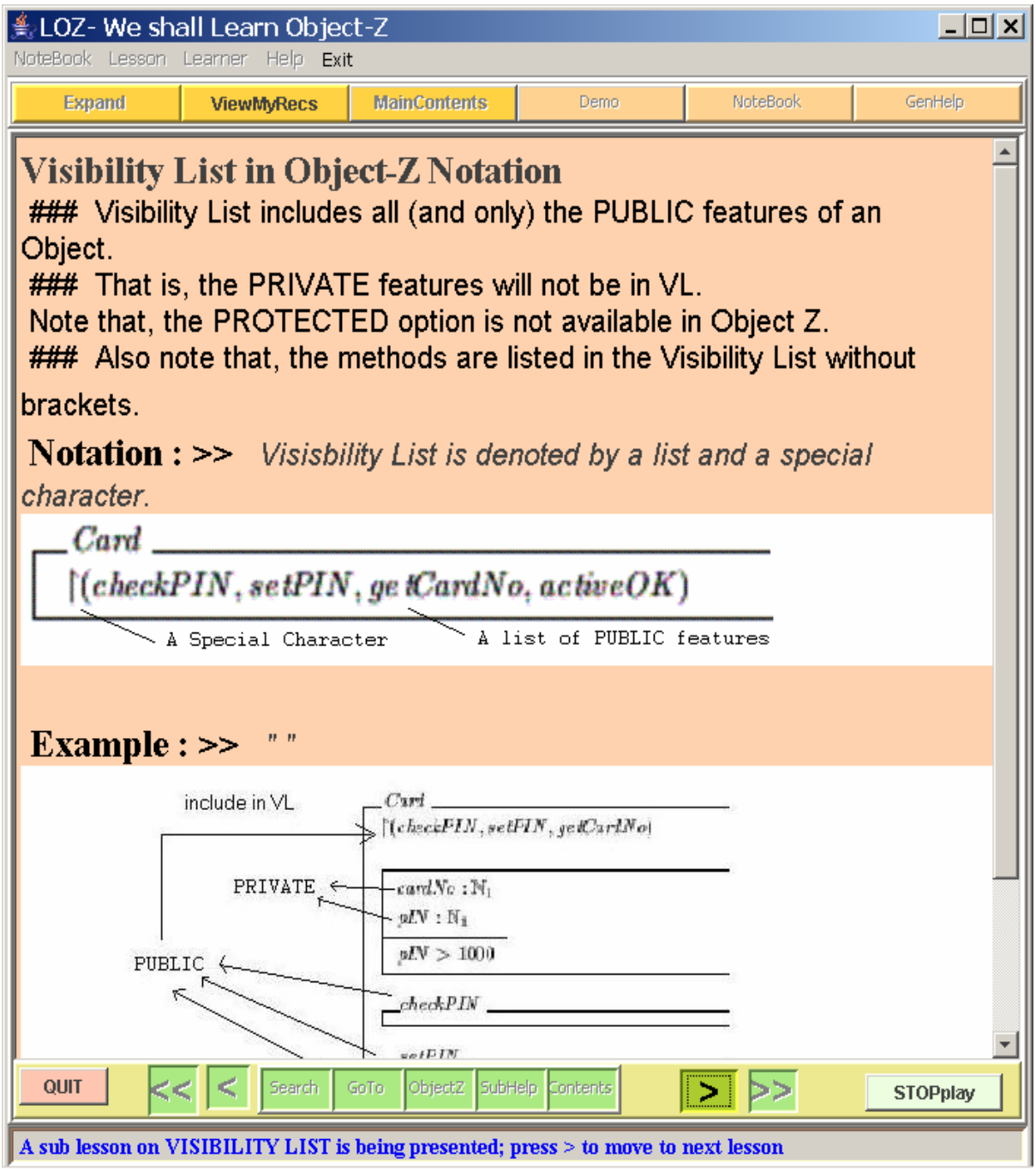


Figure B.6 Sub-Lesson 1 on Visibility List

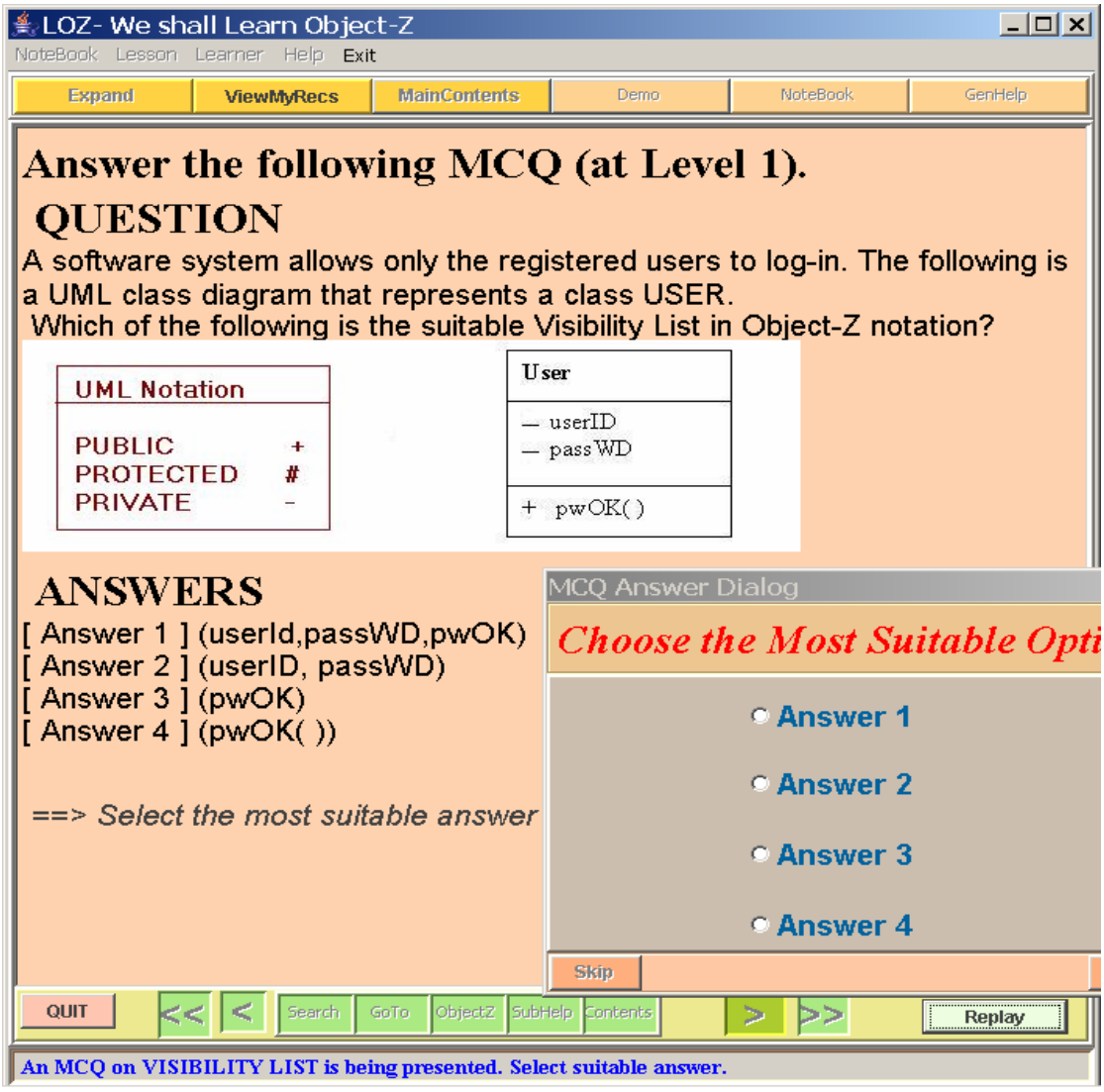


Figure B.7 MC-test at Level 1 for Sub-Lesson 1

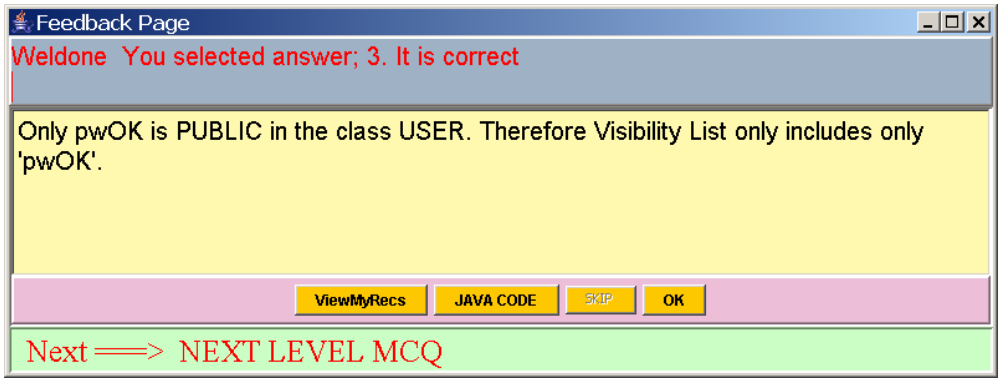


Figure B.8 Feedback for correct answer, if estimate is high

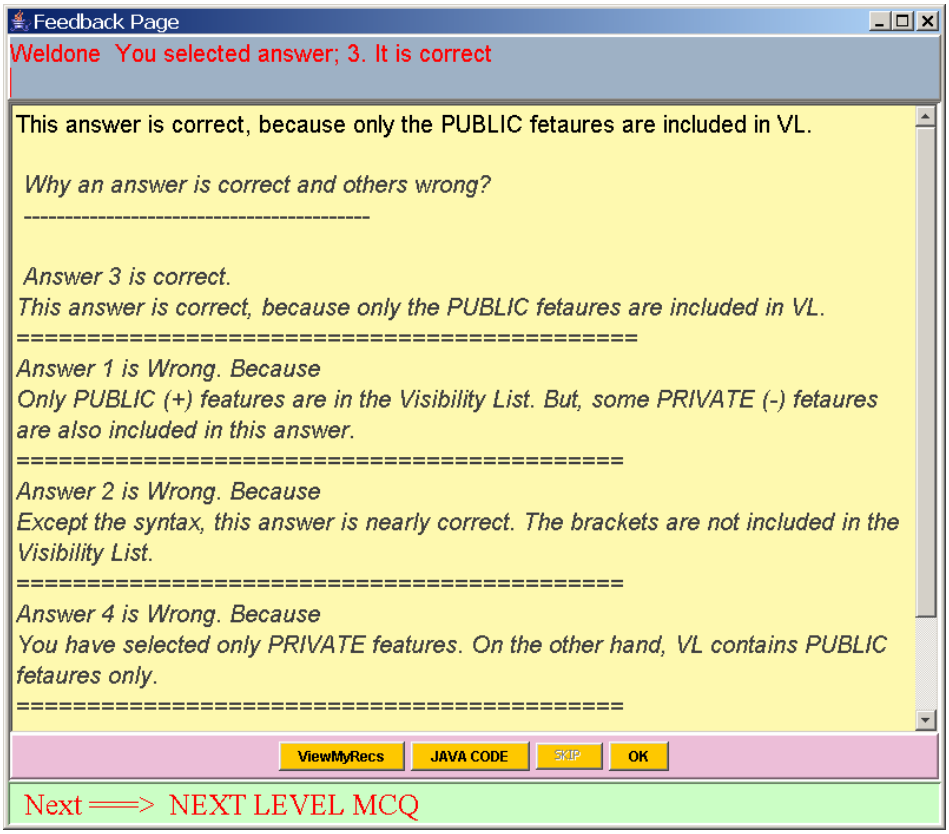


Figure B.9 Feedback for correct answer, if estimate is low

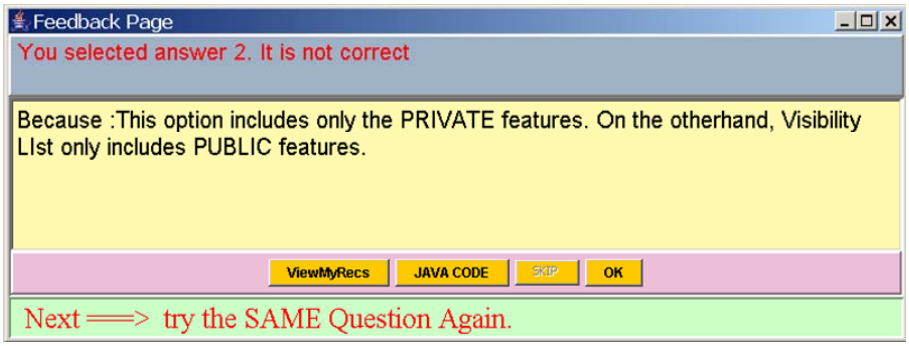


Figure B.10 Feedback for incorrect answer, if estimate is high

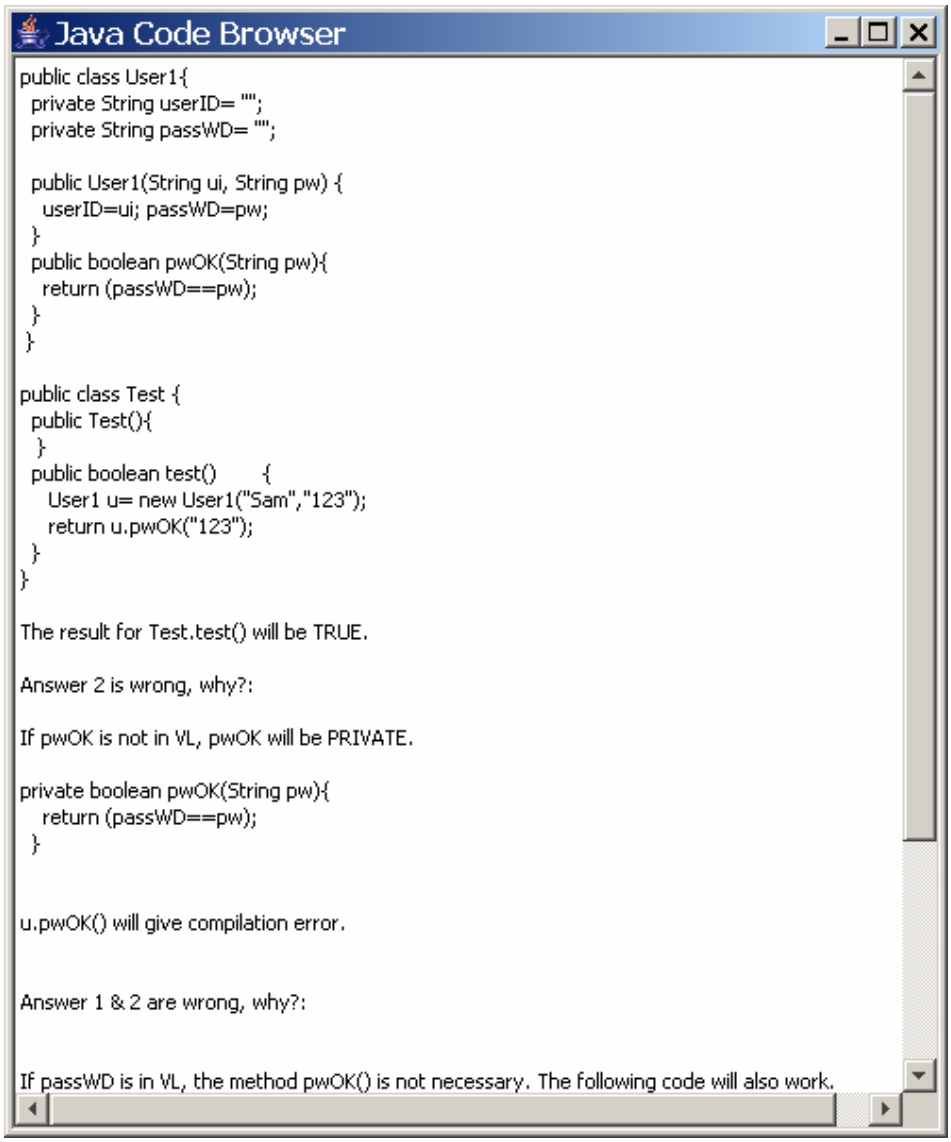


Figure B.11 Java Code Help

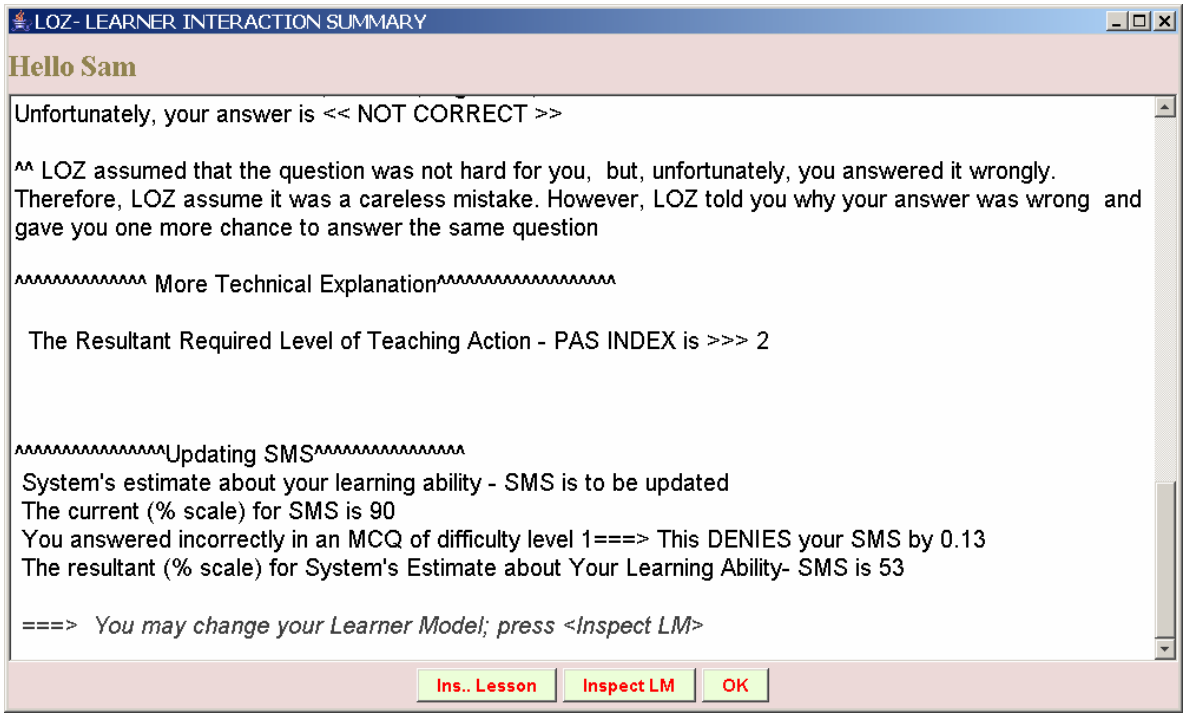


Figure B.12 Updating Learner Model

After the second and third level MC tests (Figures B.13 and B.14), the next sub-lesson (Figure B.15) is displayed. There are three MC test levels for this lesson also (Figures B.16 and B.17). Finally, there are two exploration lessons (Figure B.18 and B.19). There are some practice MC tests included for each exploration lesson (Figure B.20 and B.21).

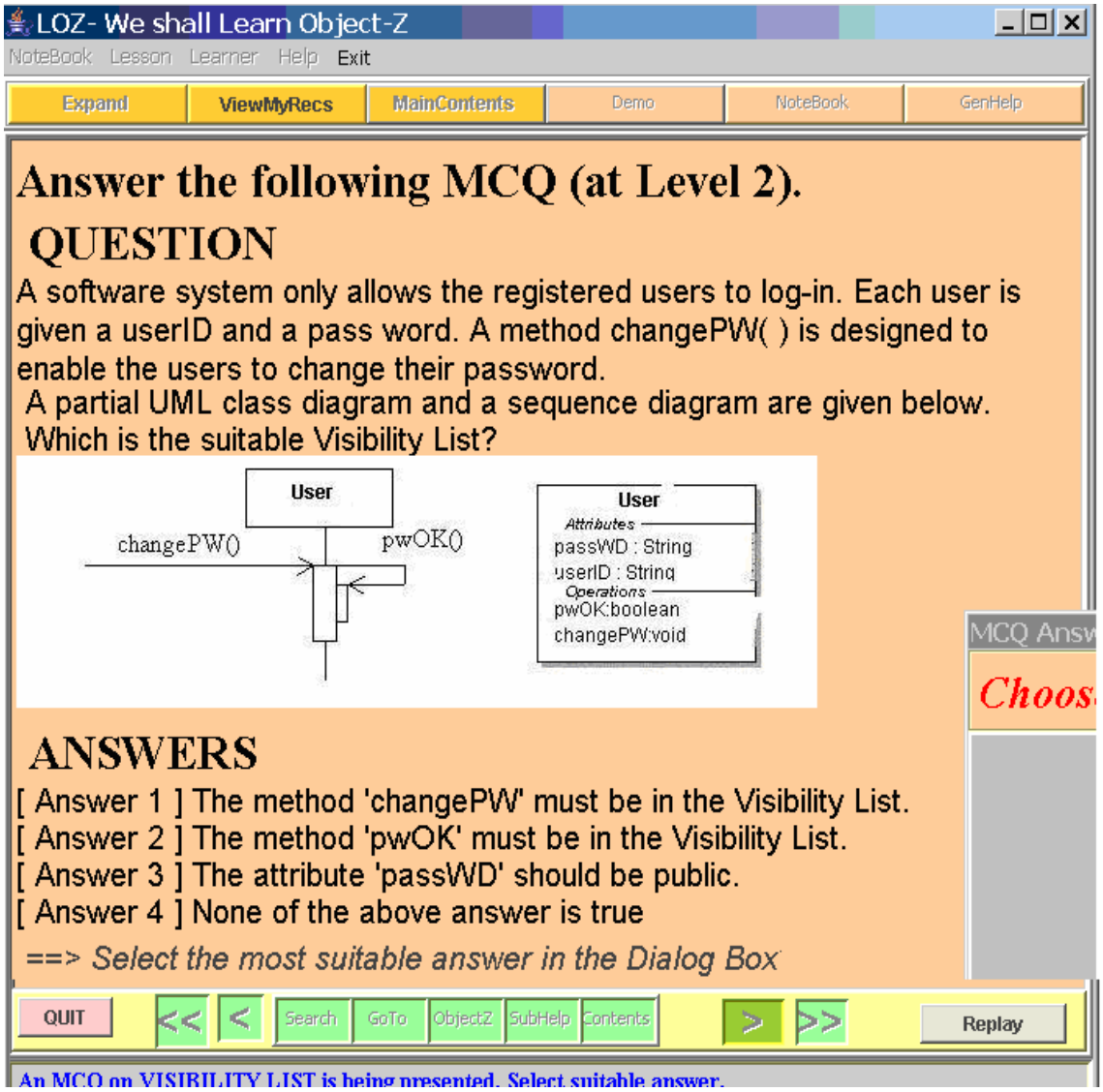


Figure B.13 Sub-Lesson 1 - MC Test at Level 2

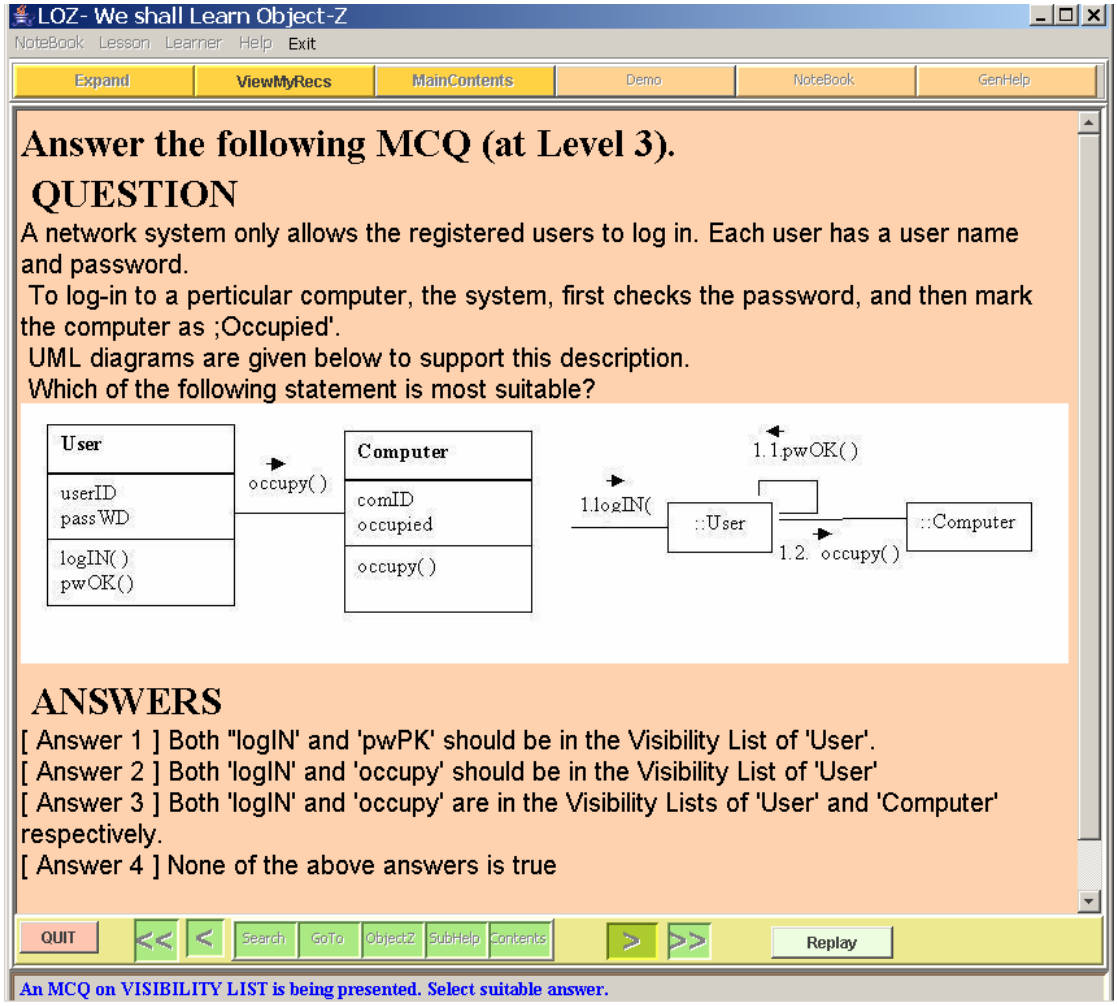


Figure B.14 Sub-Lesson 1 – MC Test at Level 3

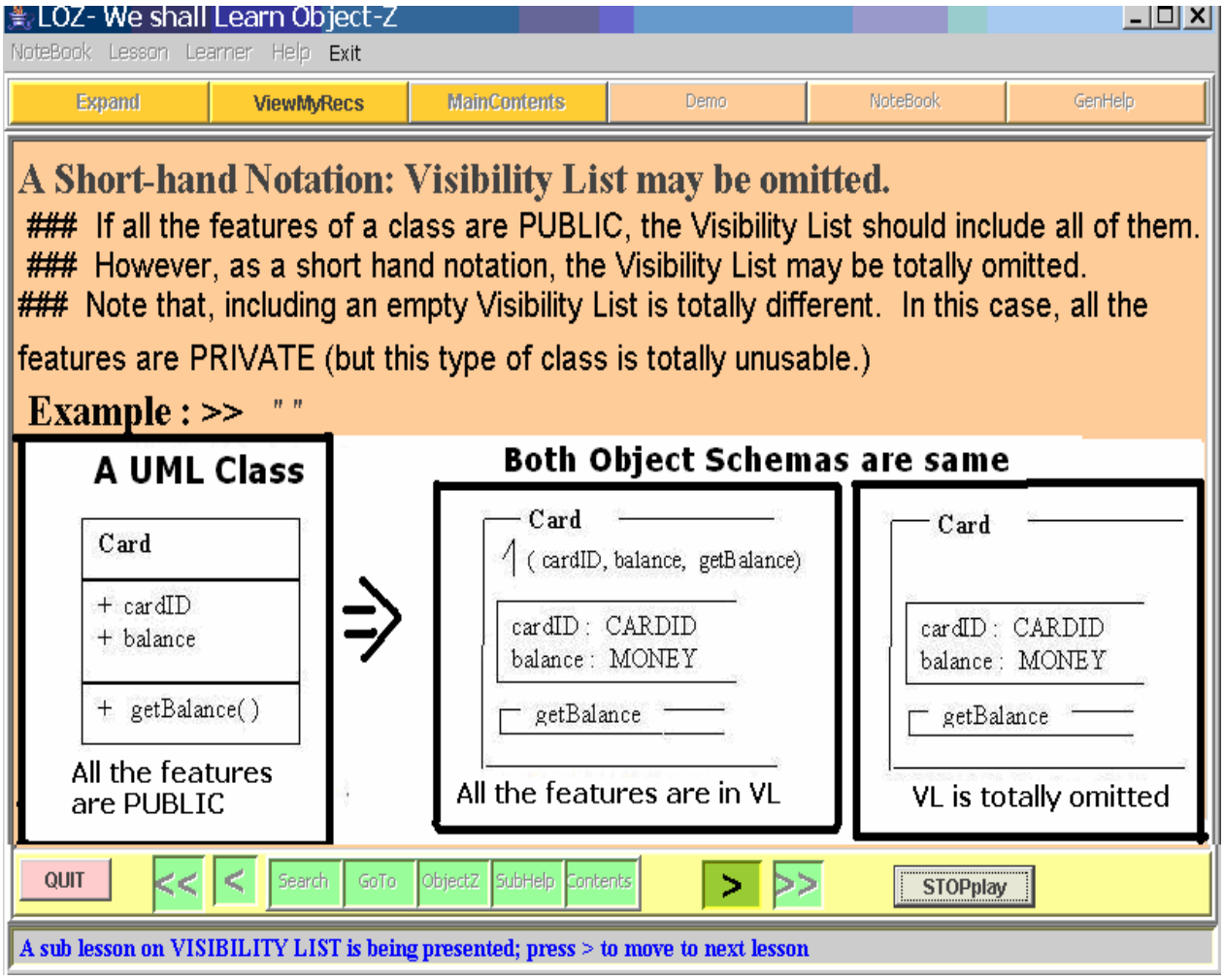


Figure B.15 Sub-Lesson 2 on Visibility List

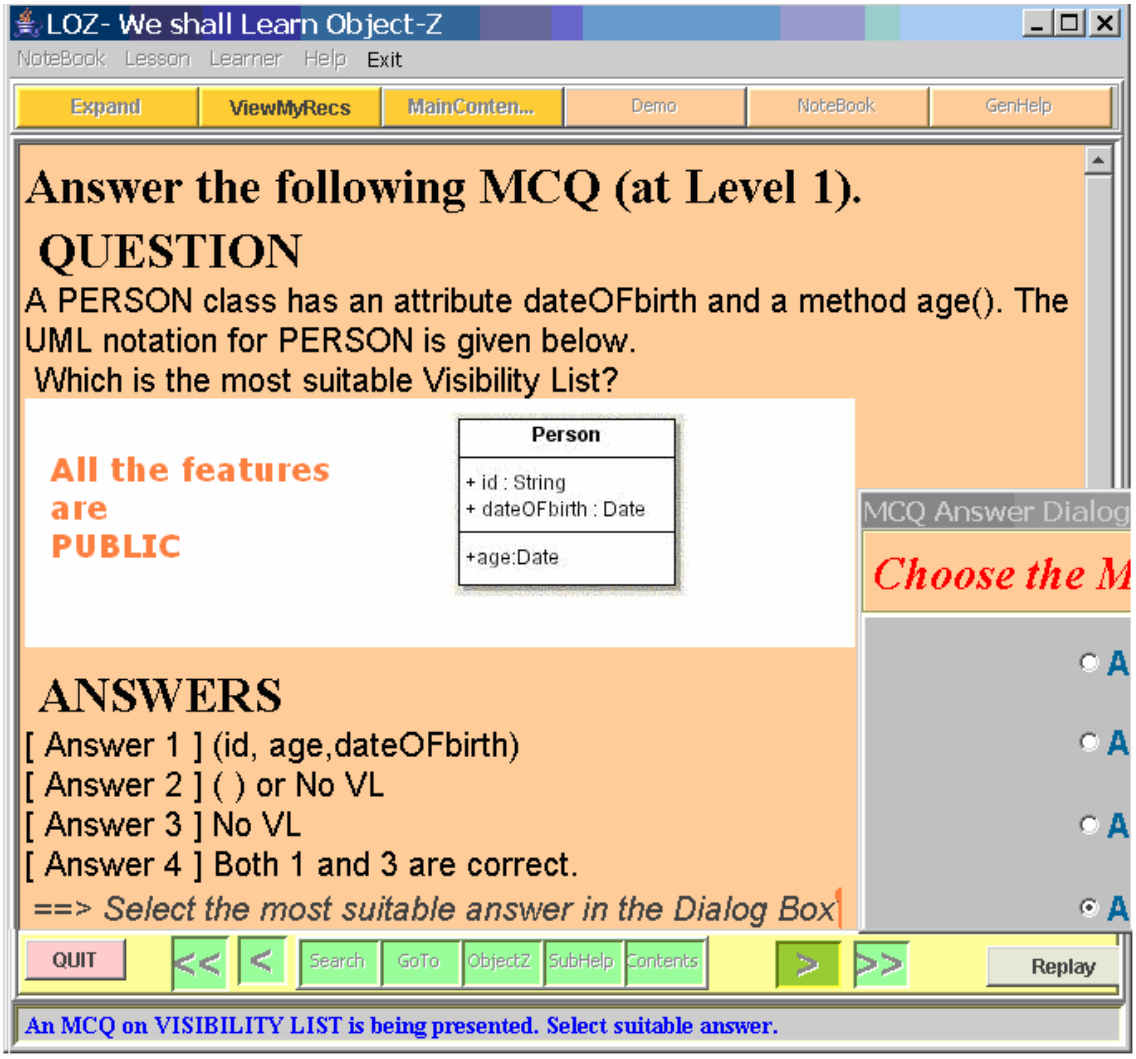


Figure B.16 Sub-Lesson 2 – MC Test at Level 1

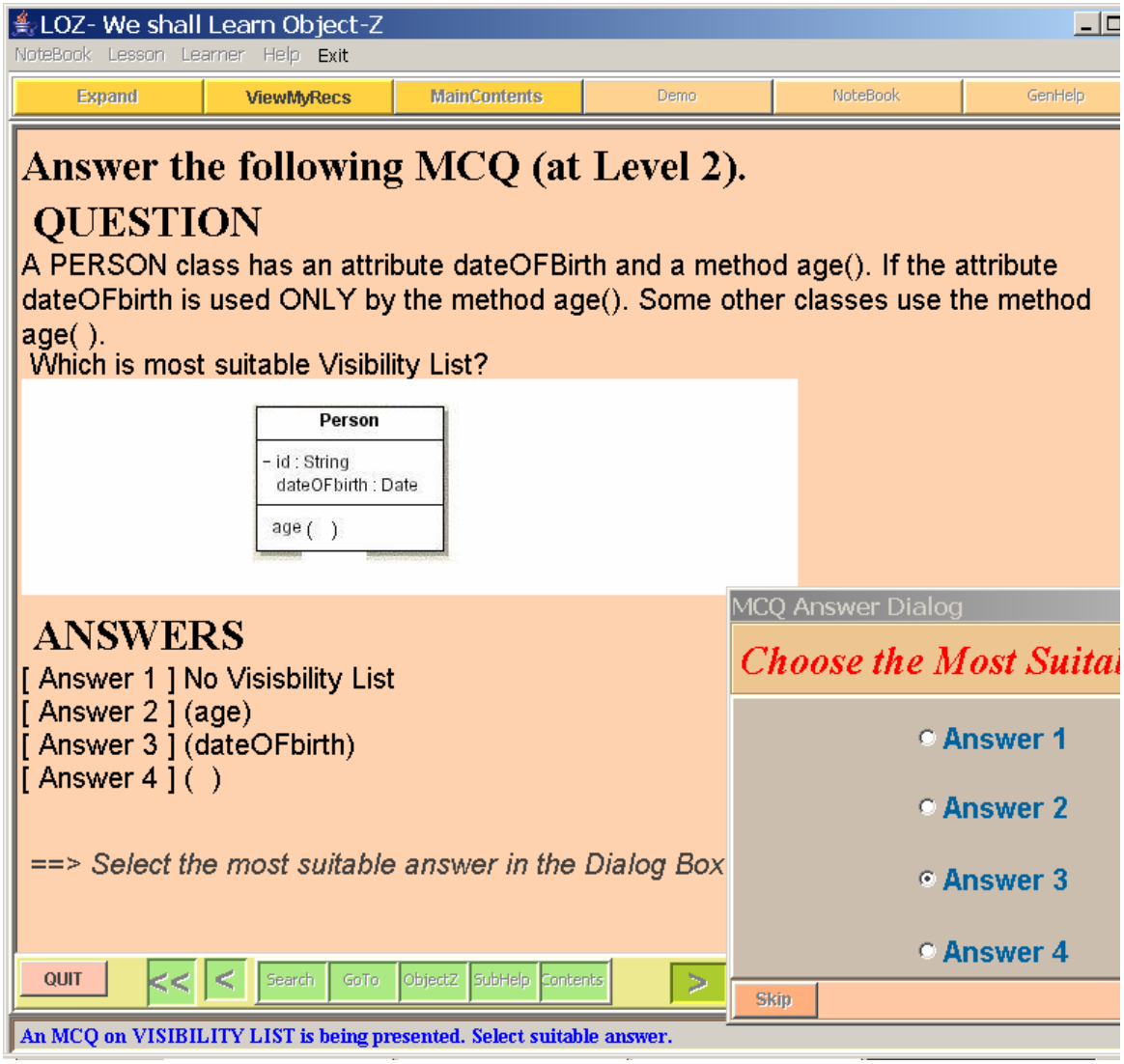


Figure B.17 Sub-Lesson 2 – MC Test at Level 2

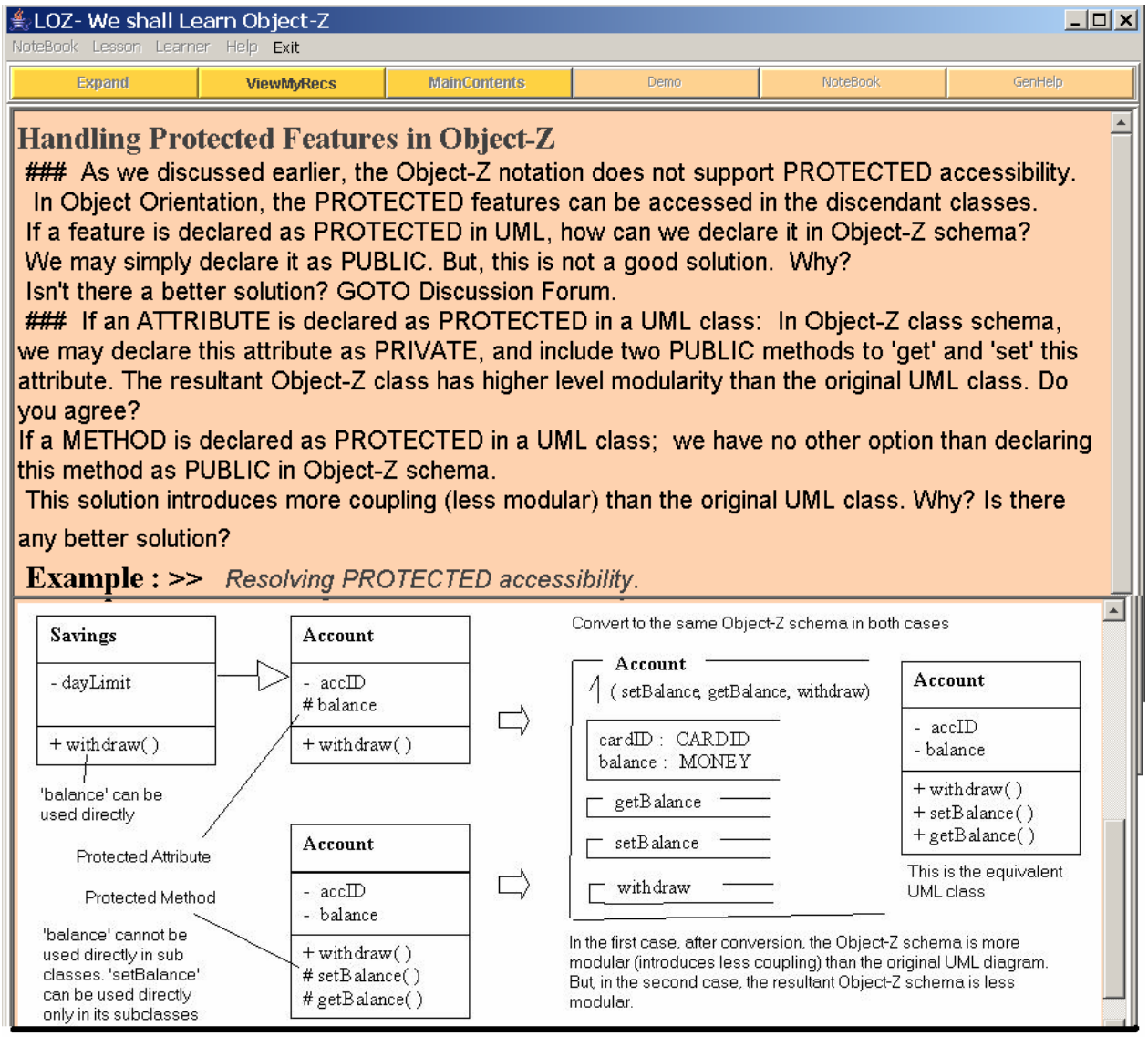


Figure B.18 Exploration Topic 1 – Visibility List

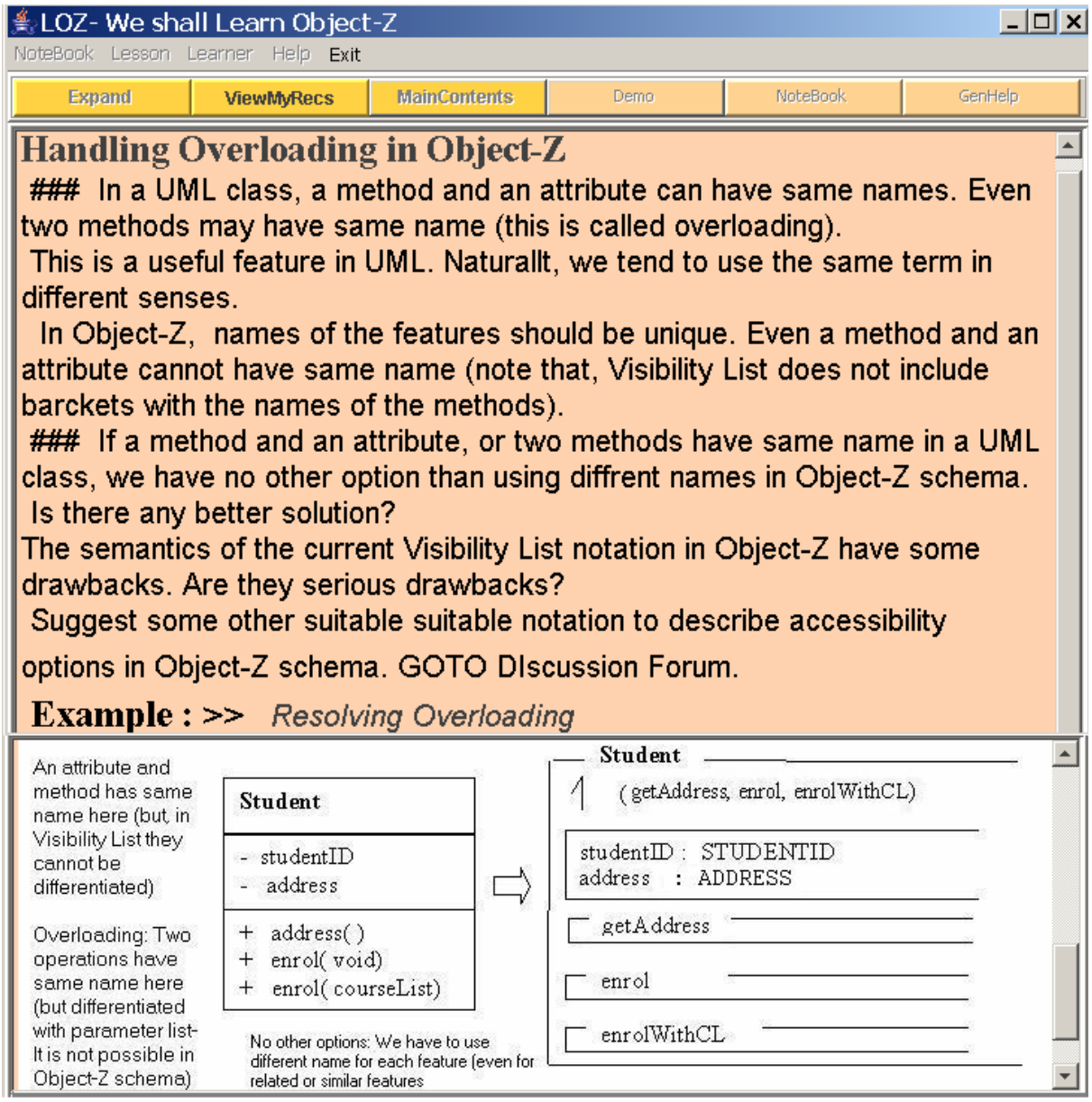


Figure B.19 Exploration Topic 2- Visibility List

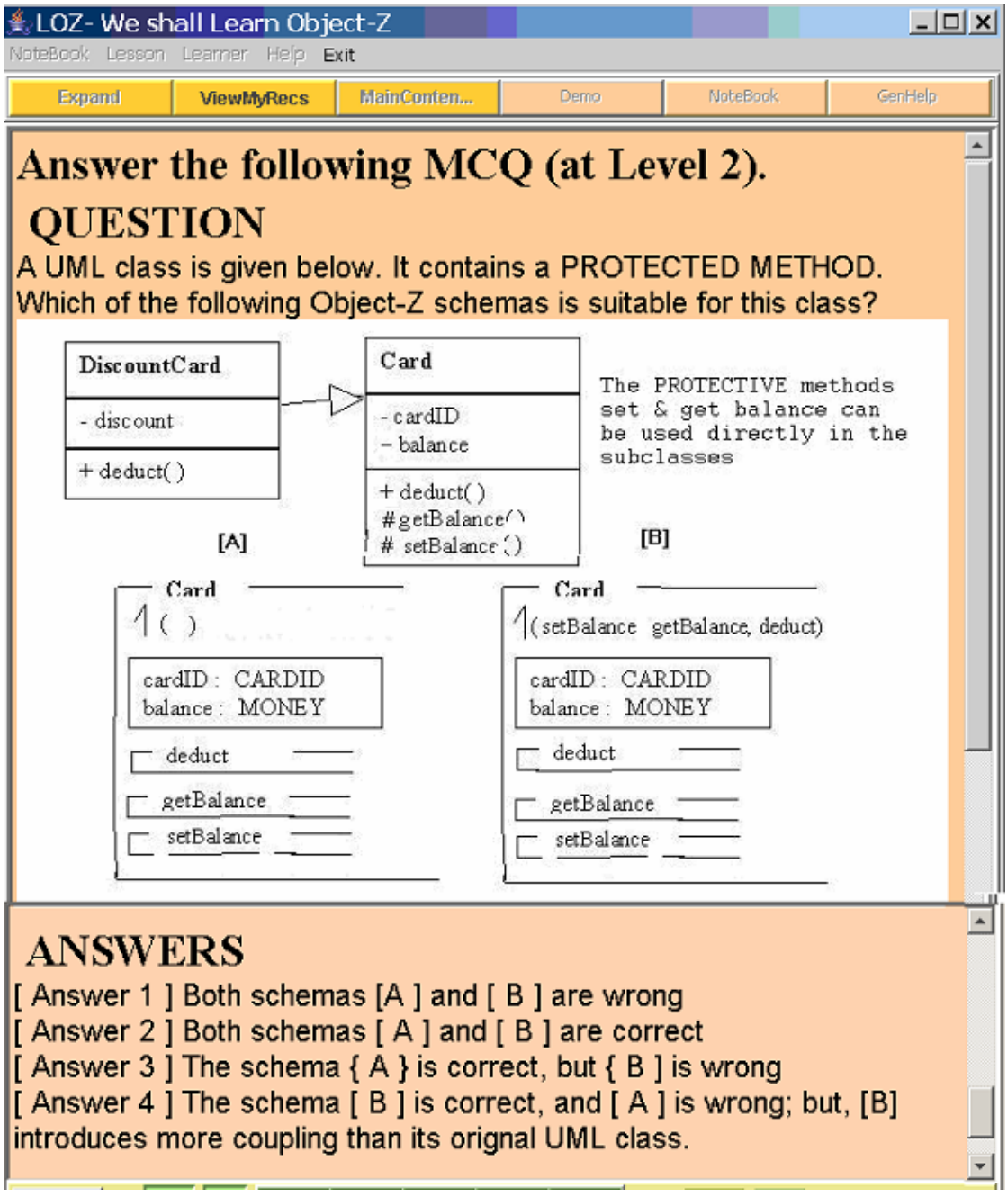


Figure B.20 Exploration Topic 1 – MC Test Level 2

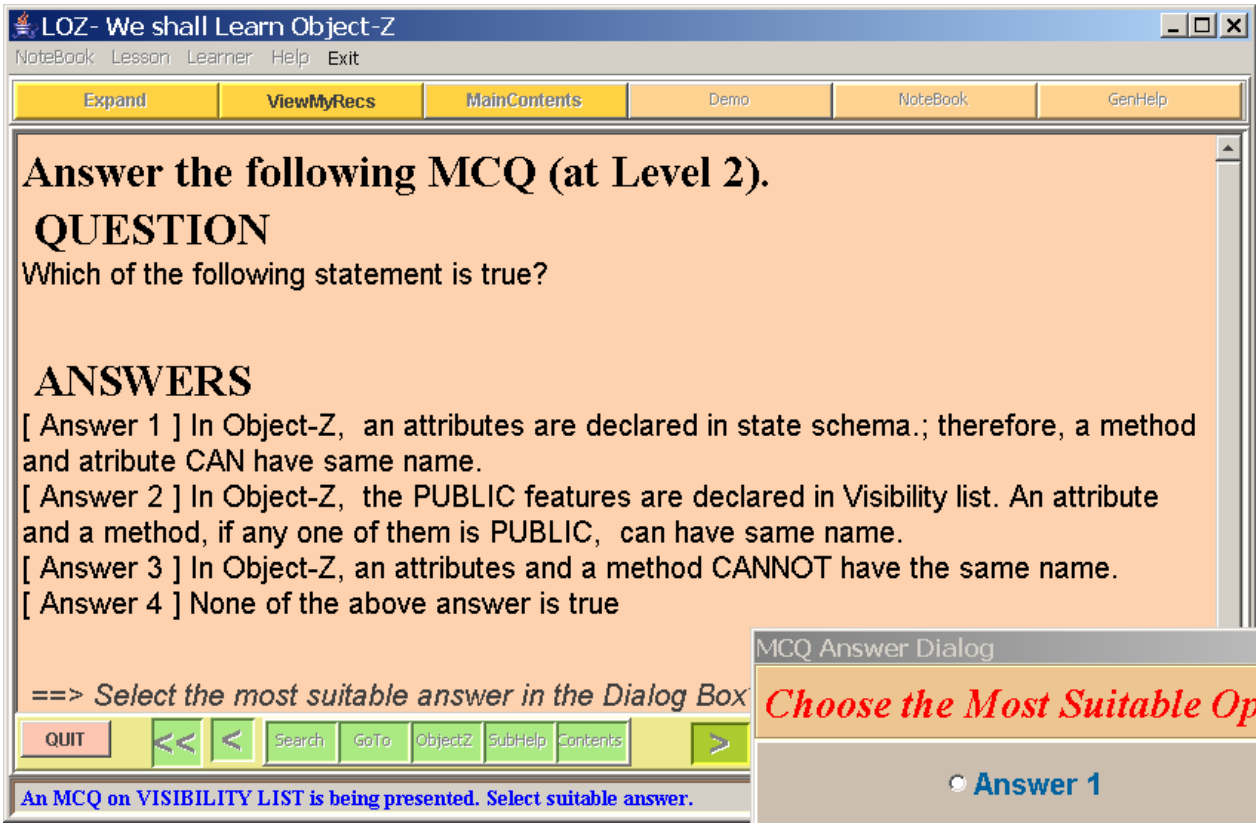


Figure B.21 Exploration Topic 2 – MC Test Level 2

Appendix C: Prototype Code

The names of the classes that are used in the prototype are given in Figure C.1 in the alphabetical order. The code for the classes are given in the attached CD under the directory named Programs.

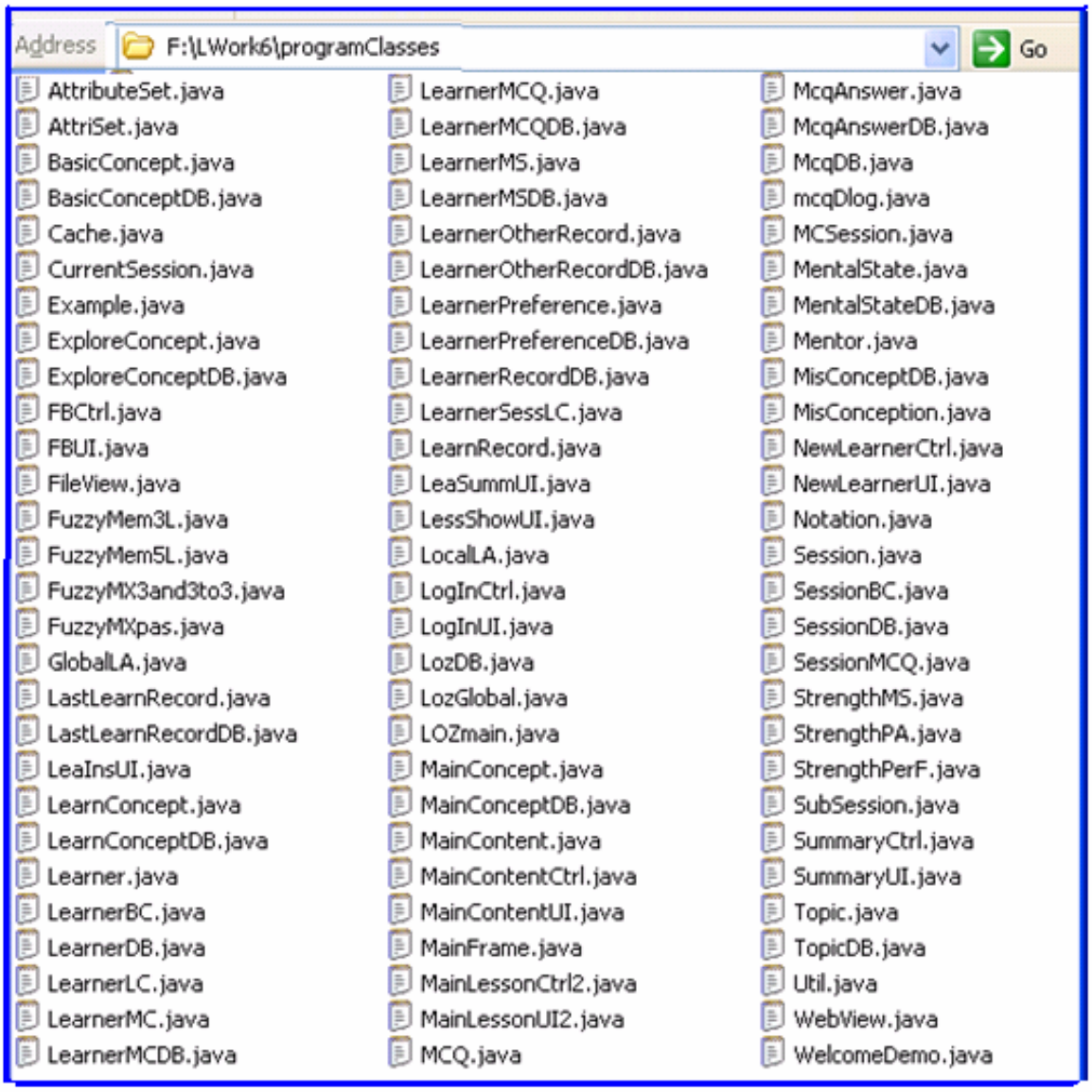
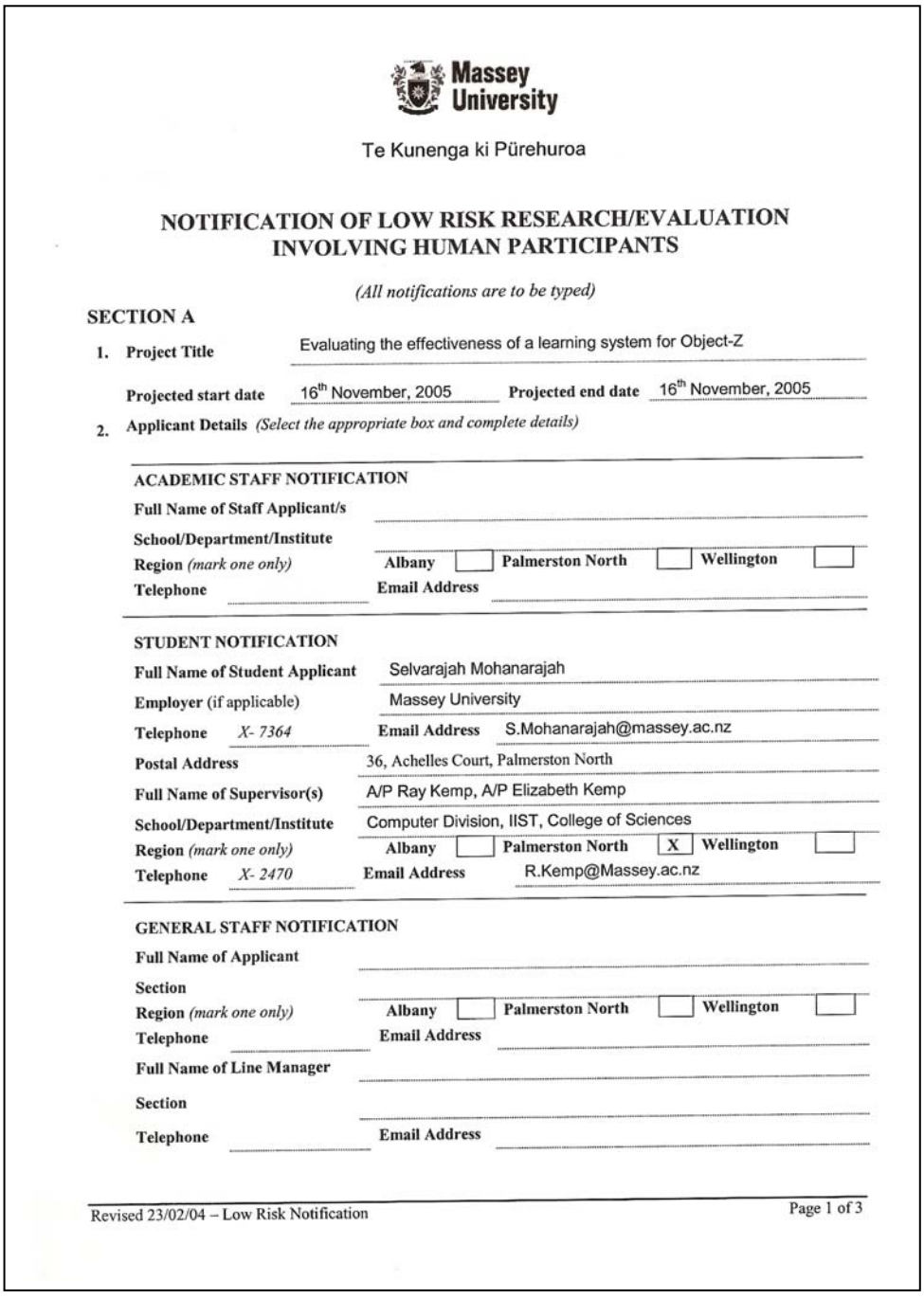


Figure C.1 Classes in the Prototype

Appendix D: Final Evaluation

D.1 Ethical Issues: Application & Approval



Massey University

Te Kunenga ki Pūrehuroa

NOTIFICATION OF LOW RISK RESEARCH/EVALUATION INVOLVING HUMAN PARTICIPANTS

(All notifications are to be typed)

SECTION A

1. **Project Title** Evaluating the effectiveness of a learning system for Object-Z

Projected start date 16th November, 2005 **Projected end date** 16th November, 2005

2. **Applicant Details** *(Select the appropriate box and complete details)*

ACADEMIC STAFF NOTIFICATION

Full Name of Staff Applicant/s

School/Department/Institute

Region *(mark one only)* **Albany** ☐ **Palmerston North** ☐ **Wellington** ☐

Telephone **Email Address**

STUDENT NOTIFICATION

Full Name of Student Applicant Selvarajah Mohanarajah

Employer (if applicable) Massey University

Telephone *X- 7364* **Email Address** S.Mohanarajah@massey.ac.nz

Postal Address 36, Achelles Court, Palmerston North

Full Name of Supervisor(s) A/P Ray Kemp, A/P Elizabeth Kemp

School/Department/Institute Computer Division, IIST, College of Sciences

Region *(mark one only)* **Albany** ☐ **Palmerston North** ☒ **Wellington** ☐

Telephone *X- 2470* **Email Address** R.Kemp@Massey.ac.nz

GENERAL STAFF NOTIFICATION

Full Name of Applicant

Section

Region *(mark one only)* **Albany** ☐ **Palmerston North** ☐ **Wellington** ☐

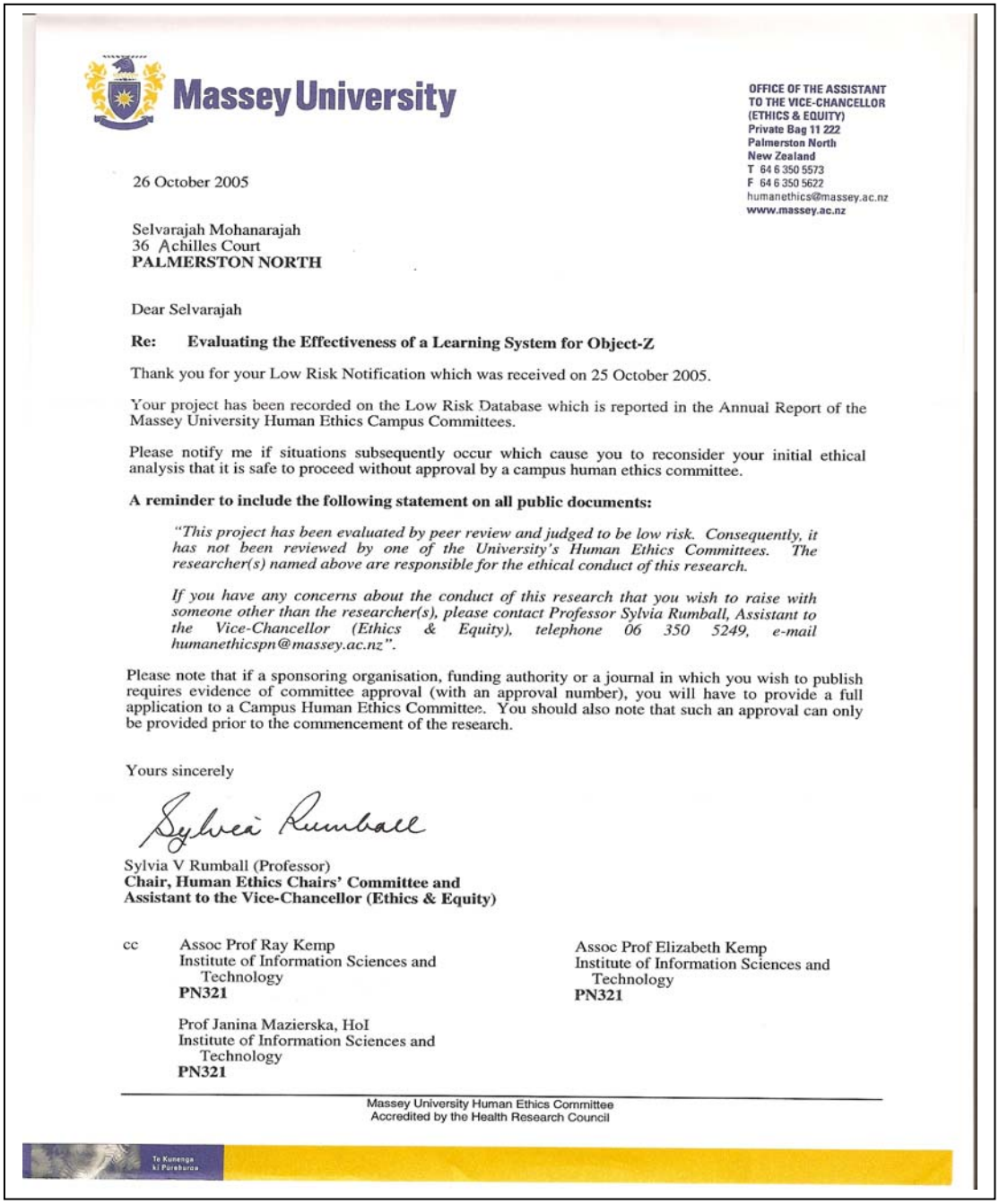
Telephone **Email Address**

Full Name of Line Manager

Section

Telephone **Email Address**

Revised 23/02/04 – Low Risk Notification Page 1 of 3



Massey University

OFFICE OF THE ASSISTANT
TO THE VICE-CHANCELLOR
(ETHICS & EQUITY)
Private Bag 11 222
Palmerston North
New Zealand
T 64 6 350 5573
F 64 6 350 5622
humanethics@massey.ac.nz
www.massey.ac.nz

26 October 2005

Selvarajah Mohanarajah
36 Achilles Court
PALMERSTON NORTH

Dear Selvarajah

Re: Evaluating the Effectiveness of a Learning System for Object-Z

Thank you for your Low Risk Notification which was received on 25 October 2005.

Your project has been recorded on the Low Risk Database which is reported in the Annual Report of the Massey University Human Ethics Campus Committees.

Please notify me if situations subsequently occur which cause you to reconsider your initial ethical analysis that it is safe to proceed without approval by a campus human ethics committee.

A reminder to include the following statement on all public documents:

> *"This project has been evaluated by peer review and judged to be low risk. Consequently, it has not been reviewed by one of the University's Human Ethics Committees. The researcher(s) named above are responsible for the ethical conduct of this research.*
>
> *If you have any concerns about the conduct of this research that you wish to raise with someone other than the researcher(s), please contact Professor Sylvia Rumball, Assistant to the Vice-Chancellor (Ethics & Equity), telephone 06 350 5249, e-mail humanethicspn@massey.ac.nz".*

Please note that if a sponsoring organisation, funding authority or a journal in which you wish to publish requires evidence of committee approval (with an approval number), you will have to provide a full application to a Campus Human Ethics Committee. You should also note that such an approval can only be provided prior to the commencement of the research.

Yours sincerely

Sylvia Rumball

Sylvia V Rumball (Professor)
Chair, Human Ethics Chairs' Committee and
Assistant to the Vice-Chancellor (Ethics & Equity)

cc Assoc Prof Ray Kemp
Institute of Information Sciences and Technology
PN321

Assoc Prof Elizabeth Kemp
Institute of Information Sciences and Technology
PN321

Prof Janina Mazierska, HoI
Institute of Information Sciences and Technology
PN321

Massey University Human Ethics Committee
Accredited by the Health Research Council

Te Kunenga ki Pūrehuroa

D.2 Information Sheet

EVALUATING A COMPUTER BASED
LEARNING SYSTEM FOR OBJECT-Z

Information Sheet

Research being conducted by Mr.S.Mohanarajah under the supervision of Associate Professor Ray Kemp and Associate Professor Elizabeth Kemp with a view of designing a computer based learning system for Object-Z notation. A learner model, which uses artificial intelligent techniques, is incorporated with the system. A software prototype has been developed to test the design. This prototype will be used in this evaluation process.

Initially, participants will be asked to answer a set of multiple-choice questions. Afterwards, they will be asked to use the prototype to learn certain Object-Z concept. Finally, they will be given another set of multiple-choice questions. Additionally, they will also be given a questionnaire to complete. The entire process will take nearly four hours. The data will be analysed to determine the effectiveness of the system for learning Object-Z notation. The quantitative data will be aggregated.

All the data collected will be anonymous. No information being collected which would identify a participant in the study. Results will appear in a thesis and academic publications. Quotations from participant's answers may also be used. The data will be stored for future reference.

Neither grades nor academic relationship with the institution will be affected by either refusal or agreement to participate. The participants have the following rights.

- Refuse to participate
- Refuse to answer any particular question
- Withdraw from the study
- Ask any question about the study during participation
- When it is concluded, to get the summary of the project findings.

This project has been evaluated by peer review and judged to be low risk. Consequently, it has been reviewed by one of the Universities human ethics committees. The researcher(s) named above are responsible for the ethical conduct of this research.

D.3 Consent Form

EVALUATING A COMPUTER BASED
LEARNING SYSTEM FOR OBJECT-Z

PARTICIPANT CONSENT FORM

This consent form will be held for a period of five (5) years

I have read the Information Sheet and have had the details of the study explained to me. My questions have been answered to my satisfaction, and I understand that I may ask further questions at any time.

I agree to participate in this study under the conditions set out in the Information Sheet.

Signature: ..... **Date:** .....

Full Name - printed .....

D.4 MC Tests: Versions 1 & 2

General Explanation on Confidence Based MC test scheme

Consider MC tests with 4 options and only one correct answer (or most suitable answer). In traditional tests only one option can be selected, but in Confidence Based MC tests, you may indicate different levels of confidence on each option. Consider the example below. There are two interfaces; traditional and confidence-based. In the confidence-based interface, initially, all the options are considered to be in 100% wrong state (Not Possible). You may indicate different level of confidence on each option. If you leave an option unmarked, similar to traditional tests, it will be assumed that you are 100% confident that the option is wrong (Not Possible).

For example;

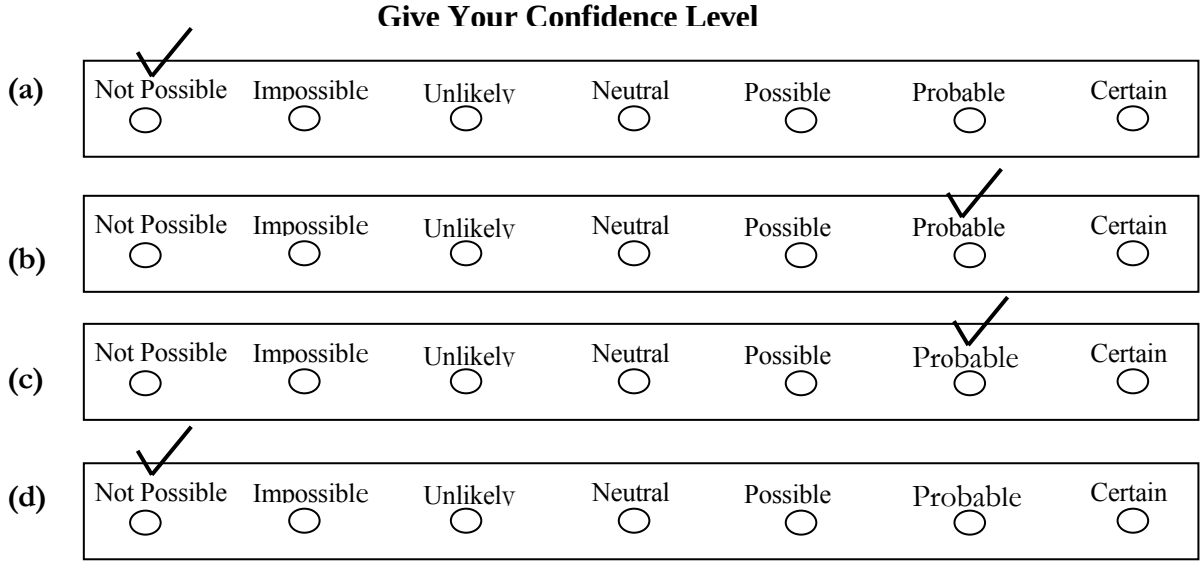
Q. The capital of Australia is:

a. Melbourne
b. Sydney
c. Canberra
d. Brisbane

Give the Most Suitable Option

c

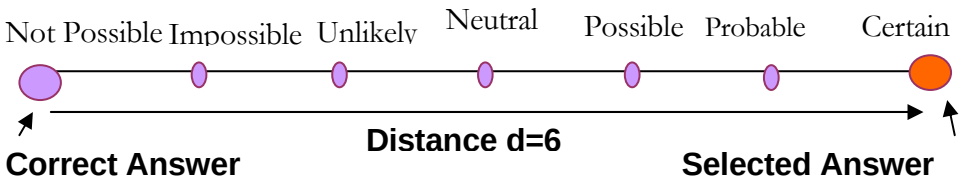
Give Your Confidence Level



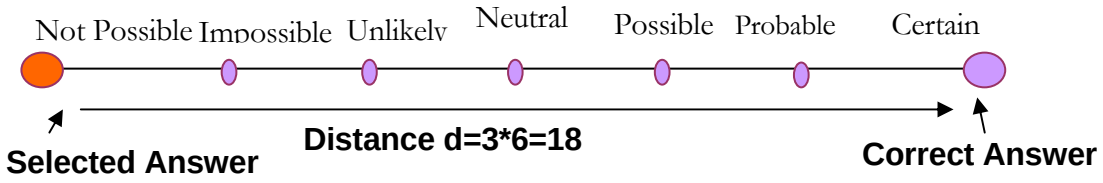
In this case, a student was sure that either b or c was correct (and also he was 100% sure that both a and d were incorrect). In the traditional MC test method, he/she had to choose between b and c. Depending on their luck, they may get full mark or nothing. As in this example, if a student marked c he will not get any mark - his partial knowledge is not rewarded. In the confidence based MC test, however, they may express their knowledge at different levels. In the above example, the student will get some reasonable marks for his answer. According to the marking scheme given below, he will get 12 (out of 18). Note that, if more than one option can be correct, two options may be marked as certain.

Marking schema:

Consider a MC test with one correct option and three distracters. There are 7 scales in the confidence-based answering interface (from Not Possible to Certain). Therefore, 7^4 possible combinations are possible. However, some selections are not practical (for example, all Not Possible or more than one Certain). We shall use a metric D that measures the distance between the actual answer and selected answer.



Consider the above illustration. If only one answer is correct among four answers, it shows the distance (here is 6) when a incorrect answer is marked as certain (100% correct).



On the other hand, if a correct answer is marked as 100% incorrect, the distance will be 18 (3*6). This is because the ratio, correct answer to incorrect answer, is 1:3.

We add these distances for each option and subtract it from 18 to get a possibly positive metric for performance [$\mathbf{D} = 18 - (\mathbf{d_1} + \mathbf{d_2} + \mathbf{d_3} + \mathbf{d_1})$]. This metric varies from -18 to +18. One will get -18 for extremely wrong beliefs. If one selects all 'not possible' (100%

wrong), or all 'certain' (100% right) or all 'not sure' gets 0. One will get 18 for correct answer. If an option is unmarked it will be considered as 'not possible'. Therefore, if a student leaves all the options unmarked for a question, he/she will get zero. All the negative marks will be considered zero, as a result, all the students will get some positive marks.

MC Test (Ver 1) - Visibility List

Instruction:
There are **ten** questions. Each has **four** options; where, only **one** is correct.
First, for all the questions, indicate the most suitable option in the box provided.
Additionally, for all the questions, please indicate your confident levels **for all the options**.

Q(1)

Visibility List is used in Object-Z to specify:

a. Inheritance Hierarchy of a Class
b. Overloaded methods of a class
c. Accessibility options of the features of a class
d. None of the above

Give Your Confidence Level

	Not Possible	Impossible	Unlikely	Neutral	Possible	Probable	Certain
(a)	○	○	○	○	○	○	○
(b)	○	○	○	○	○	○	○
(c)	○	○	○	○	○	○	○
(d)	○	○	○	○	○	○	○

Q(2)

An educational institution issues magnetized identity cards to control access to computer labs. The following UML class is designed.

Account
- accID - balance
+ withdraw( )

Which is the visibility list in the related Object-Z schema?

a. ↾ (accID, balance, withdraw)

b. ↾ (accID, balance)

c. ↾ (withdraw( ))

d. ↾ (withdraw)

Give Your Confidence Level

(a) Not Possible ○ Impossible ○ Unlikely ○ Neutral ○ Possible ○ Probable ○ Certain ○

(b) Not Possible ○ Impossible ○ Unlikely ○ Neutral ○ Possible ○ Probable ○ Certain ○

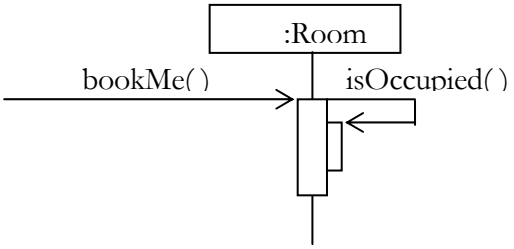
(c) Not Possible ○ Impossible ○ Unlikely ○ Neutral ○ Possible ○ Probable ○ Certain ○

(d) Not Possible ○ Impossible ○ Unlikely ○ Neutral ○ Possible ○ Probable ○ Certain ○

Note: The confidence-based answering interface is given for each question as in question (1) and (2) just after the answering option. Hereafter, in order to save space it will not be included hereafter.

Q(3)

A hotel has several rooms. A room may be single or double. A room may be occupied or vacant. The following extracts from UML diagrams are available. The names for the methods and attributes are self explanatory.



Room
- roomNo - bed // double or single occupied // true or false

isOccupied( )// returns 'occupied'
bookMe( )
releaseMe( ) // occupied ← false

Which one of the following statements is true?

a. The method 'bookMe' may not be in the Visibility List.
b. The method 'isOccupied' must be in the Visibility List.
c. The method 'bookMe' must be in the Visibility List.
d. The attribute 'occupied' must be in the Visibility List.

Give Your Confidence Level

Q(4)

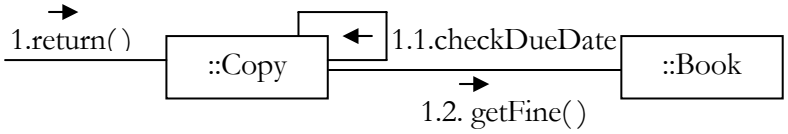
If a Visibility List is not included in an Object-Z schema;

a. All the features of the class should be private
b. All the features of the class should be public
c. All the feature of the class should be protected
d. All the methods should be public, but the attributes should be private

Give Your Confidence Level

Q(5)

Consider the following extract from a collaboration diagram. When a book is returned to a library, first, a clerk scans the book to get the identity codes (bookID and copyID). If the due-date is passed, the system should display the required fine.



Which of the following statements is true?

a. Both methods 'return' and 'getFine' should be in the Visibility List of the class Copy.
b. Both methods 'return' and 'checkDueDate' should be in the Visibility List of the class Copy.
c. Both methods 'return' and 'getFine' should be in the Visibility Lists of the classes Copy and Book respectively.
d. All of the above may be false.

Give Your Confidence Level

Q(6)

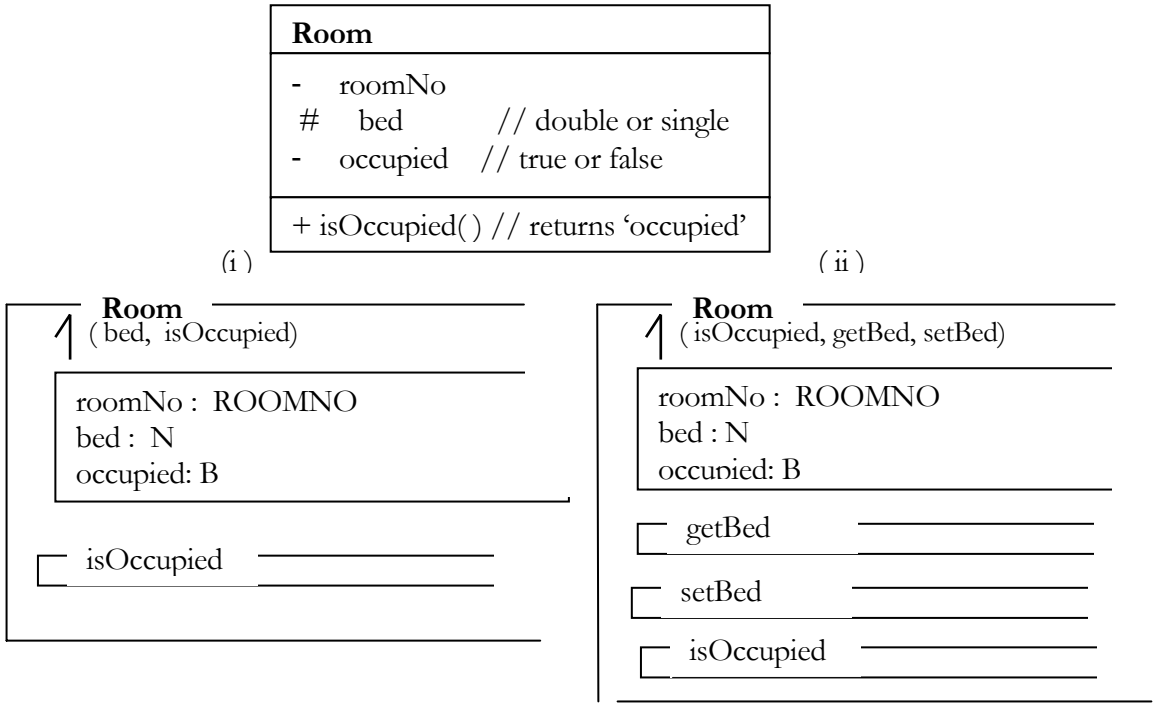
If the Visibility List in an Object-Z schema is an empty set; which of the following statements is true?

a. Since it is empty, the Visibility List may not be included in the Object-Z schema.
b. All the features of the class will be private, and therefore the class will not be usable.
c. All the features of the class will be public, however, the class may introduce unnecessary coupling.
d. None of the above is true

Give Your Confidence Level

Q(7)

Consider the following UML class. Note that, it contains a protected attribute 'bed'. Protected attributes are not supported in Object-Z. Two Object-Z schemas, (i) and (ii), are designed to represent the class 'Room'. All the names are self-explanatory.



Which of the following statements is true?

a. (i) is correct, but (ii) is incorrect
b. (ii) is correct, but (i) is incorrect
c. Both are correct, but (ii) is more modular.
d. Both are correct, but (i) is more modular.

Give Your Confidence Level

Q(8)

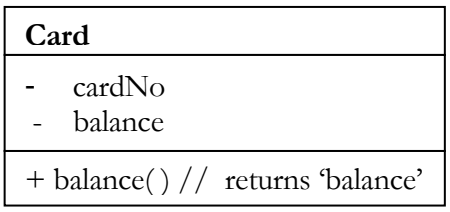
A method X is defined as 'protected' in a class. Which of the following is true?

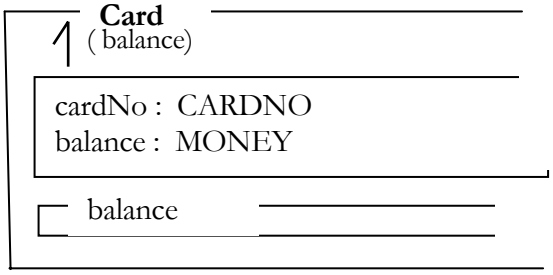
a. X should be in the Visibility List
b. X may or may not be in the Visibility List
c. If X returns a value, it should be in the Visibility List
d. None of the above true.

Give Your Confidence Level

Q(9)

A member of your group designed the following Object-Z schema 'Card' for the given UML class 'Card'.





What is wrong in the Object-Z schema?

a. The visibility list should contain 'balance( )' not 'balance'
b. Both 'balance( )' and 'balance' should be included in the Object-Z schema.
c. Two features cannot have the same name in an Object-Z schema.
d. None of the above is true

Give Your Confidence Level

Q(10)

Which of the following is true in both UML classes and Object-Z schemas?

a. An attribute cannot be public.
b. At least one feature must be private.
c. A method may be private
d. Methods can be overloaded.

Give Your Confidence Level

Visibility List – MC Test (Ver 2)

Instruction:

There are **ten** questions. Each has **four** options; where, only **one** is correct.

First, for all the questions, indicate the most suitable option in the box provided.

Additionally, for all the questions, please indicate your confident levels for all the options.

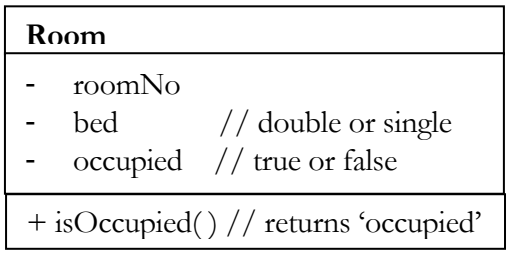
Q(1)

A Visibility List in Object-Z schema includes:

a. Private features of a class
b. Public features of a class
c. Sub classes of a class
d. None of the above

Q(2)

A hotel has several rooms. A room may be single or double. A room may be occupied or vacant at any time. The following UML class is designed.



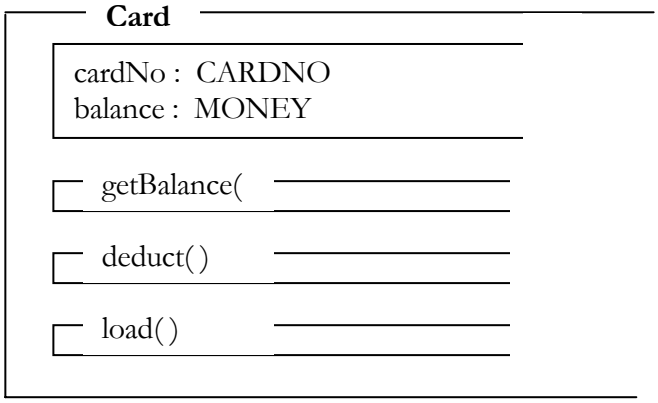
Which is the possible visibility list?

a. ↾ (occupied, roomNo, bed)
b. ↾ (roomNo, bed)
c. ↾ (isOccupied( ))
d. ↾ (isOccupied)

Give Your Confidence Level

Q(3)

An educational institution issues magnetized identity cards to its students. These cards may be used for a number of facilities such as photocopying. The following Object-Z schema is designed. According to the requirements, the methods 'getBalance', 'deduct', and 'load' are sufficient to manipulate the attribute 'balance'. The names used in this schema are self -explanatory. The visibility list needs to be included.



Which of the following statement is true?

a. The operation 'getBalance', and the attribute 'balance' should be in the Visibility List.
b. If the operations 'getBalance', 'deduct', and 'load' be in the Visibility List, the attribute 'balance' may be private.
c. If the attribute 'balance' is in the Visibility List, the methods 'getBalance', 'deduct', and 'load' may be declared in some other class.
d. (b) and (c) are correct

Give Your Confidence Level

Q(4)

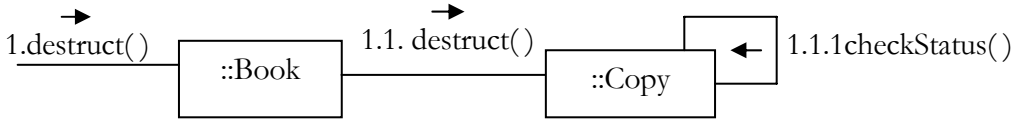
If all the features of a class are public, which of the following statements about the corresponding Object-Z schema is true?

a. The visibility list should contain all the features.
b. The visibility list can be totally omitted
c. The visibility list should be an empty set
d. Both a and b are correct

Give Your Confidence Level

Q(5)

Consider the following extract from a collaboration diagram. In order to discard a book all of its copies should be deleted before from the system. A copy can be removed only if its status is 'not-in-use'.



Which of the following statements is true?

a. No two features can have same name in an Object-Z schema. Therefore, one of the 'destruct' methods should be renamed.
b. Both methods 'destruct' and 'checkStatus' should be in the Visibility List of the class Copy.
c. Both classes Book and Copy will have their own methods 'destruct' in their respective Visibility Lists.
d. All of the above statements are false.

Give Your Confidence Level

Q(6)

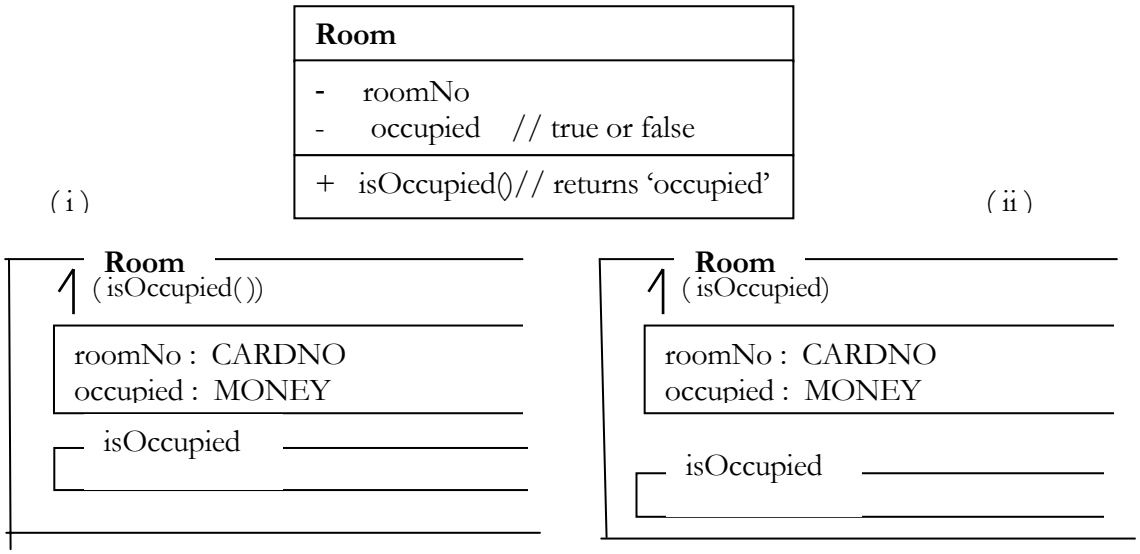
If all the features are private for a class, the Visibility List of the corresponding Object-Z schema will be,

a. An empty set, therefore, it can be excluded from the Object-Z schema
b. An empty set, however the class is not usable
c. The set of all the features
d. None of the above true

Give Your Confidence Level

Q(7)

Consider the following UML class. Two Object-Z schemas, (i) and (ii), are designed for this class.



Which of the following statement is true?

a. (i) is correct, but (ii) is incorrect
b. (ii) is correct, but (i) is incorrect
c. Both are correct, but (ii) is more modular.
d. Both are incorrect

Give Your Confidence Level

Q(8)

An attribute X is 'protected' in a UML class. But, protected attributes are not possible in Object-Z. Therefore, the following two solutions are suggested for conversion.

(i) Include X be in the Visibility List; therefore, all other classes (including all sub classes) can freely access it.

(ii) Declare X as private and include two public methods, 'set' and 'get' to set and get the value of X respectively.

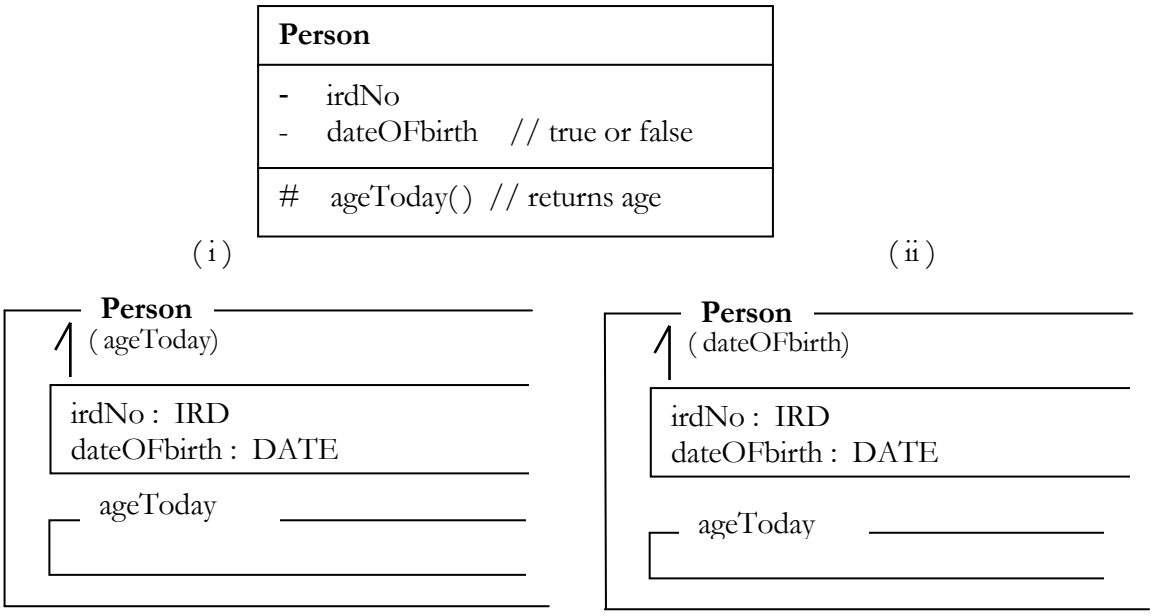
Which of the following is correct?

a. (i) is correct, but (ii) is incorrect
b. (ii) is correct, but (i) is incorrect
c. Both are correct, but (i) may introduce more coupling.
d. Both are correct, but (ii) may introduce more coupling.

Give Your Confidence Level

Q(9)

Consider the following UML class. Note that, it contains a protected method 'ageToday'. Protected methods are not supported in Object-Z. Two Object-Z schemas, (i) and (ii), are designed to solve this issue. All the names are self-explanatory.



Which of the following statement is correct?

a. Both are correct, but (ii) may give more coupling.
b. Both are correct, but (i) may give more coupling.
c. Both are incorrect
d. None of the above are correct

Give Your Confidence Level

Q (10).

Methods can be overloaded in UML classes. Which of the following is true?

a. As in UML, the parameter lists can be used to distinguish overloaded methods in the visibility list of an Object-Z schema.
b. If only one of the overloaded methods is public, method overloading is possible in Object-Z schemas.
c. Method overloading is not possible in Object-Z schemas, because, no two features can have same name in an Object-Z schema.
d. None of the above are correct.

Give Your Confidence Level

D.5 Questionnaire on LOZ

Survey Date: **November 16, 2005**

Instruction: **This survey has 13 statements. Each has five options. Please tick only one box. Please answer all the items.**

---

(1) UML support (Pre- requisite lessons)

Before introducing a topic in Object-Z, a related topic is presented to the learners.

For example: Before explaining Visibility List, The system explains about UML accessibility options (public, private and protected features)

Statement: This feature is useful for learning Object-Z notation.

☐	☐	☐	☐	☐
Strongly disagree	disagree	Neutral	agree	Strongly agree

(2) Step-by step learning (Scaffolding)

While learning a new Object-Z concept, initially, relevant UML diagrams are given; and gradually, the UML support is withdrawn.

For example: While learning Visibility List, the system gives relevant UML class diagram with accessibility options (public, private and protected features). In the next step, the class diagram is given without accessibility option; and finally, even the class diagram will not be given.

Statement: This feature is helpful for learning complex Object-Z concepts.

☐	☐	☐	☐	☐
Strongly disagree	disagree	Neutral	agree	Strongly agree

(3) Exploring, rather than cramming

Instead of merely explaining the current Object-Z notation, the system encourages the learners to criticize the existing features.

For example: Object-Z does not allow overloading. But, overloading is natural to human beings. The system encourages the learner to suggest alternative notations.

Statement: This feature is useful for learning in general.

☐	☐	☐	☐	☐
Strongly disagree	disagree	Neutral	agree	Strongly agree

(4) Feedback

The system tries to estimate the learners' state of knowledge and gives feedback according to that.

For example: If you make a careless mistake, the system will not bombard you with a detailed explanation; instead, it will just allow you to try again.

Statement: The system gives appropriate feedback.

☐	☐	☐	☐	☐
Strongly disagree	disagree	Neutral	agree	Strongly agree

(5) Java Code

For an Object-Z schema, the system may give resultant code in Java.

Statement: This facility helps to boost the motivation to learn Object-Z.

☐	☐	☐	☐	☐
Strongly disagree	disagree	Neutral	agree	Strongly agree

Opening Learner Model

(6) Viewing System's estimate

As stated before, the system tries to estimate the learners' state of knowledge and gives feedback according to that. The system may under or over estimate you. You may check the system's estimate.

Statement: This facility is very useful as I can understand how the system rate my learning ability.

☐	☐	☐	☐	☐
Strongly disagree	disagree	Neutral	agree	Strongly agree

(7) Modifying System's estimate

As explained in item (7), if you feel that the system does not give appropriate feedback to you, the system allows you to modify the estimate.

Statement: This facility is very useful as I can control the system's estimate about myself.

☐	☐	☐	☐	☐
Strongly disagree	disagree	Neutral	agree	Strongly agree

(8) Modifying the required feedback, instead of System's estimate.

As explained in item (8), the system allows you to modify the system's estimate about your ability. Instead, you may also select the required level of feedback for a certain question, this will, in turn; modify the system's estimate about your ability

Statement: As a learner, I can easily identify my required level of feedback than proposing an own estimate for my learning ability.

☐	☐	☐	☐	☐
Strongly disagree	disagree	Neutral	agree	Strongly agree

(9) Modifying the mentor model

As explained in item (2), the system explains an Object-Z concept in a step-by-step fashion using UML models.

The system allows you to start at a higher level and/or change the steps as you wish.

Statement: This facility is very useful as I can take control over my learning process.

☐	☐	☐	☐	☐
Strongly disagree	disagree	Neutral	agree	Strongly agree

Confidence Based Multiple-choice tests

(10) Expressing Partial Knowledge

Confidence based multiple choice tests allow you to express partial knowledge in a test item.

Statement: The given confidence based multiple-choice test format allows me to express my partial knowledge easily.

☐	☐	☐	☐	☐
Strongly disagree	disagree	Neutral	agree	Strongly agree

(11) Comparing with traditional (Multiple-Choice) MC tests.

Statement: I prefer the given confidence based multiple-choice test format over the traditional multiple-choice testing method?

☐	☐	☐	☐	☐
Strongly disagree	disagree	Neutral	agree	Strongly agree

General Issues

The system is only prototype with limited functionality. It is not robust and inefficient. The interface is not pleasing. Please keep these issues in mind before making your selection on the following statements.

(12) General Experience

Statement: I think the system gives suitable environment to learn Object-Z concepts.

☐	☐	☐	☐	☐
Strongly disagree	disagree	Neutral	agree	Strongly agree

(13) Satisfaction

Once the system is **fully** developed, it may be used as a revision tool for Object-Z course.

Statement: If the system is **fully** developed, I will recommend this system to my friends.

☐	☐	☐	☐	☐
Strongly disagree	disagree	Neutral	agree	Strongly agree

D.6 Raw Scores on MC Tests

Student No	Sub Group	Score on Pre Test	Score on Post Test	Score on Ver- 1	Score on Ver-2
1	1	5	9	9	5
2	1	6	9	9	6
3	1	4	8	8	4
4	1	4	10	10	4
5	1	8	10	10	8
6	2	1	4	1	4
7	2	4	8	4	8
8	2	2	8	2	8
9	2	4	9	4	9
10	2	7	9	7	9

There were two versions of the test. Both had 10 questions each. There were two groups of students. Both groups had 5 members each. The first group was given version-1 of the test for the pre-test and version-2 for the post-test. But for the second group, version-2 was given for the pre-test and version-1 for the post-test. The number of correct answers for each student is given in the above table.

D.7 Responses to the Questionnaire

There are ten students participated in this survey. The percentages given in the last two rows refer to the responses in the last two (agree) or first two points (disagree) in the scale respectively.

Question Number		No. of Responses for each Likert Level					SD	Avg	Agree (%)	Disagree (%)
		one	two	three	four	five				
Sec1	1	-	-	-	7	3	0.48	4.3	100	0
	2	-	--	1	5	4	0.67	4.3	90	0
	3	-	-	6	3	1	0.71	3.5	40	0
	4	-	-	1	5	4	0.67	4.3	90	0
	5	-	1	4	5	-	0.70	3.4	50	10
Sec2	6	-	1	2	5	2	0.92	3.8	70	10
	7	-	1	5	4	-	0.67	3.3	40	10
	8	-	4	2	4	-	0.94	3	40	40
	9	-	-	2	8	-	0.42	3.8	80	0
Sec3	10	-	-	-	2	8	0.42	4.8	100	0
	11	-	1	-	4	5	0.95	4.3	90	10
Sec4	12	-	-	1	6	3	0.63	4.2	90	0
	13	1	-	-	6	3	1.15	4	90	10

Appendix E: Fuzzification Mechanism for Prediction

The following example illustrates the performance prediction mechanism explained in the Section 8.3.2.

Assume that the SMS of a learner is 65. Suppose he/she is going to answer a question with difficulty (D=35). In the fuzzification process, according to the fuzzy membership function given in figure 6.2, the relevant degree of memberships can be determined:
For Strength of Mental State (SMS):

$$\text{SMS}= 65 \rightarrow \mu_{Strong}(65) = 0.25,\ \mu_{Medium}(65) = 0.75;$$

And , for Difficulty (D):

$$\text{D}= 35 \rightarrow \mu_{Low}(35) = 0.25,\quad \mu_{Moderate}(65) = 0.75$$

The relevant fuzzy rules are given in the rows of Table 8.1
For example, the first row represents the following rule:

IF SMS is Strong AND
D is Low
THEN
Learner will answer the question CORRECTLY

Similarly the rules can be formulated from the other rows.

By using the following fuzzy AND rule: $\mu_{X\ AND\ Y}(c) = \min(\mu_X(c), \mu_Y(c))$.

In general, in a complex if-then rule using AND in the condition part, the confidence of the necessary part will be given by the minimum value of the confidences in the individual statements that make up the condition.

In the rule propagation, the following rules give the intermediate values for degree of memberships for Performance (P);

SMS(Strong) & D(Low) → P(Correct)

That is, $\mu_{correct}(C) = \min(0.25, 0.25) = 0.25,$

SMS(Strong) & D(Moderate) → P(Correct)

That is, $\mu_{correct}(C) = \min(0.25, 0.75) = 0.25,$

SMS(Medium) & D(Low) → *P(Correct)*

That is, $\mu_{correct}(C) = min(0.75, 0.25) = 0.25$,

SMS(Medium) & D(Moderate) → *P(Medium)*

That is, $\mu_{medium}(C) = min(0.75, 0.75) = 0.75$,

The resultant degrees of memberships for the variable P are;

$$\mu_{correct}(C) = max(0.25, 0..25, 0.75) = 0.75$$

$$\mu_{medium}(C) = 0.75$$

$$\mu_{incorrect}(C) = 0.0$$

Finally, in the defuzzification process, the crisp value C of P can be determined using the fuzzy membership function given in figure 6. However, at this juncture, the learner model just need to predict the performance. In other words, the learner model needs to select a range rather than a single numerical value for P. Therefore, in defuzzification, it is sufficient to select the P level (the linguistic value) with the highest degree of membership. If two P levels have equal beliefs the P with the highest strength will be selected.

We have,

$$\max(\mu_{correct}(C), \mu_{medium}(C)) = max(0.75, 0.75)$$

The highest degree of membership is 0.75, and the corresponding fuzzy value is *Correct* or *Medium.* The least extreme value may be selected.

Therefore, the required P level will be, *Medium*

That is, if a learner with the Mental State (SMS=65) failed a question with difficulty level (D=35), the system expects medium level performance. That is, in a traditional MC test scheme, the answer may be either correct or incorrect.