UNIVERSITY OF SOUTHAMPTON

FACULTY OF ENGINEERNG, SCIENCE AND MATHEMATICS

School of Electronics and Computer Science

Usability of Semi-formal and Formal Methods Integration – Empirical Assessments

by
Rozilawati Razali

A doctoral thesis submitted in partial fulfilment
of the requirements for the award of
Doctor of Philosophy

March 2008

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Rozilawati Razali

Software practitioners are provided with an enormous choice of methods and tools for improving software quality. They seem to adopt a new technology only if there is convincing evidence that the technology is usable. Furthermore, it is no longer acceptable in software engineering to claim that a new technology is usable without accompanying scientific evidence. Empirical assessments provide an ideal mechanism for evaluating software technology. As a single assessment can never embrace all possible situations, it is reasonable to acquire some evidence of a new technology's likely efficacy when used under certain conditions.

The use of formal notations such as B increases a model's precision and consistency. However, the notations are regarded as being difficult to comprehend due to unfamiliar symbols and underlying rules of interpretation that are not so apparent to practitioners. Semi-formal notations such as Unified Modelling Language (UML) use graphical representations to present system elements. They therefore are perceived as more accessible. Nevertheless, such notations cannot be verified systematically to ensure the correctness of a model. Perhaps by combining both semi-formal and formal notations could produce a model that is not only accurate and consistent but also accessible to practitioners.

This thesis presents several empirical assessments conducted on a modelling method that integrates the use of B formal notation and the semi-formal notation of UML, namely UML-B. The main objective of the assessments is to evaluate the usability of the method. This includes the comprehensibility, learnability, operability and attractiveness of the notation used in the method and the method itself in supporting modelling process. The assessments comprise a series of controlled experiments and surveys. The controlled experiments evaluate the comprehensibility of the notation from stakeholders' perspective for model validation and maintenance tasks. On the other hand, the surveys assess the usability of the method and the supporting tools from developers' perspective for model creation tasks. The findings of the assessments suggest that the method is able to produce a comprehensible formal model. The method is accessible to users only when the principles and roles of both notations are obvious and well understood, and when there is strong support from the environment. Based on the findings, a set of usability theories of integrated methods such as UML-B has been generated. Design profiles have also been proposed, which provide future designers with some guidelines for designing usable integrated methods and verification tools. The thesis also provides practitioners with some understanding of the strengths, weaknesses, opportunities and threats of methods that integrate semi-formal and formal approaches.

Table of Contents

List of Figures

List of Tables

DECLARATION OF AUTHORSHIP

I, Rozilawati Razali, declare that the thesis entitled Usability of Semi-formal and Formal Methods Integration – Empirical Assessments and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of the thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as:

1) Razali, R. and Garratt, P. W. (2006). Measuring the Comprehensibility of a UML-B model and a B model. *Proceedings of the International Conference on Computer and Information Science and Engineering (CISE 2006),* 16, 338-343.

2) Razali, R., Snook, C. F., Poppleton, M. R., Garratt, P. W. and Walters, R. J. (2007). Experimental Comparison of the Comprehensibility of a UML-based Formal Specification versus a Textual One. *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering (EASE)*, 11, 1-11.

3) Razali, R., Snook, C. F., Poppleton, M. R. and Garratt, P. W. (2007). Usability Assessment of a UML-based Formal Modelling Method. *Proceedings of the 19th Annual Psychology of Programming Workshop (PPIG)*, 56-71. (Awarded as one of the Best Papers).

4) Razali, R., Snook, C. and Poppleton, M. (2007). Comprehensibility of UML-based Formal Model – A Series of Controlled Experiments. *Proceedings of the International Workshop on Empirical Assessment of Software Engineering Languages and Technologies (WEASELTech)*, *co-located with the 22nd IEEE/ACM International Conference on Automated Software Engineering* (*ASE*), 25-30.

5) White, P (Ed.) (2007). *Deliverable D34 (D7.4)-Assessment Report 3* Rigorous Open Development Environment for Complex Systems (RODIN), IST-511599. Available online at http://rodin.cs.ncl.ac.uk/D34.pdf (One of the contributors).

6) Razali, R., Snook, C. F., Poppleton, M. R. and Garratt, P. W. (2008). Usability Assessment of a UML-based Formal Modelling Method Using Cognitive Dimensions Framework. *Human Technology: An Interdisciplinary Journal on Humans in ICT Environments*. (In press).

Signed: ……

Date: ……

Acknowledgements

I would like to thank my supervisor, Dr. Paul Garratt for his supervision and the Malaysian Government including the National University of Malaysia for the sponsorship.

I am greatly indebted to Dr. Colin Snook and Dr. Michael Poppleton for their invaluable advice and support. In particular, Dr. C. Snook has provided technical guidance on UML-B and empirical assessments whereas Dr. M. Poppleton has provided advice and mechanisms for the execution of the empirical work. I am also thankful to the examiners, Prof. Barbara Kitchenham and Prof. Paul Lewis for their precious constructive comments.

I am very thankful to my postgraduate colleagues of DSSE: Reza, Andy, Lis, Divakar, John, Jenny, Tossaporn, KD and Ida, for their involvement in the pilot studies. The appreciation also goes to COMP3011 (Southampton) and CS389 (Surrey) students for their participation in the empirical work (Spring term 2006 and 2007). This includes the anonymous reviewers of the research materials and the "peer-reviewed" papers resulted from this research.

Last but not least, I would like to express my deepest gratitude to my family: husband, parents, parents-in-law and siblings for their continuous encouragement, prayer and support. Without them, the completion of this research might have not been possible.

Chapter 1

Introduction

1.1. Background to the Problem

Formal methods have been proposed as a part of software development lifecycle for many years. They have been applied in a wide variety of settings particularly in the development of complex and critical systems (Houston et al., 1991; McDermid, 1993; Gerhat et al., 1993; 1994; Hall, 1996; Bowen et al., 1997; Ross, 2005), where the issue of safety and security is the main priority. There are various types of formal methods available (Formal Methods Virtual Library, 2007). Several examples include VDM (Jones, 1990; Fitzgerald et al., 1998), B (Abrial, 1996) and Abstract State Machines (Gurevich et al., 2000; Börger et al., 2003). Although formal methods encompass the entire process of system development, they are widely used in the specification development.

A formal specification is a system description expressed in a notation whose vocabulary, syntax and semantics are formally defined using mathematical constructs (Sommerville, 2001). Even if formal methods are not employed beyond the analysis and design phase, a formal specification is a useful artefact as it helps improving the quality of the later product (Bowen et al., 2006). The formality imposed by the notation enables the early detection of specification errors, which are expensive to correct if they remain undetected until later stages of software development process (Boehm et al., 2001).

Despite many asserted advantages that formal methods and their artefacts could offer (Craigen et al., 1995; Hinchey, 2002; Bowen et al., 1995; 2006), there is still much debate on the practicality of the methods. One of the major concerns with formal methods is the ability of software practitioners to overcome the mathematical barriers in a formal specification. The mathematical notation used in the formal specification is always perceived as difficult to read and understand (Finney et al., 1996a; Finney, 1996b; Carew et al, 2005). Therefore, it is not surprising that the successful use of formal methods in research laboratories has had little real impact on industrial practice for day-to-day software development (Zimmerman et al., 2002).

Development methods that employ notations such as the Unified Modelling Language (UML) (OMG, 2006) mainly use abstract graphical representation for specifying system requirements. The notations possess some modelling rules and support refinement activities. However, they cannot be verified systematically to ensure the consistency and accuracy of the specification. Even if it is possible to confirm the interconnection between entities and perform syntactic checking, it is almost impossible to ensure each successive refinement's correctness. Due to this respect, the notations are considered as semi-formal.

On the other hand, the graphical representation in semi-formal notations is an asset that should not be undermined. Previous studies have shown that the representation does have significant advantages for certain tasks particularly software comprehension (Vessey et al., 1986; Cunniff et al., 1987; Scanlan, 1989; Curtis et al., 1989). It is believed that the representation is in some sense analogous to the world that it represents (Bauer et al., 1993; Stenning et al., 1995) and thus is more intuitive. Despite this claim however, researchers have yet to agree the superiority of graphical representation over textual (Petre, 1995; Blackwell et al., 2001). One of the reasons is that the underlying factors that contribute to the superiority of graphical representation are not well understood (Scaife et al., 1996). Moreover, a purely graphical representation is not as expressive as a textual representation. To claim that a graphical representation is more powerful than a textual representation therefore seems to be too ambitious. Perhaps it is better to take the view that a graphical representation plays an

important role in software comprehension as a companion to a textual representation.

Several studies have highlighted the importance of choosing a usable notation for software specification (Macaulay, 1996; Britton et al., 1999). The notation used determines the comprehensibility of a specification, which indeed is an essential communication mechanism among different stakeholders. Specifications must be comprehensible to stakeholders with diverse backgrounds and expertise, who in fact may be geographically distributed. Moreover, the efficiency of development process depends highly on the technology used. The notation that is easier to learn and use by developers is more preferable.

The choice of notation for a particular project often reflects the experience or preferences of the development team more than an objective consideration of possible alternatives (McCluskey et al., 1995). Lack of empirical evidence to support theories, models and decisions in software engineering is one of the reasons (Fenton et al., 1994; Pfleeger, 1999; Perry et al., 2000). Unless the specific factors that cause a software technology to be more or less effective are understood, its adoption will continue to be a random act (Perry et al., 2000). Researchers therefore must learn to produce evidence that is useful for practitioners. Software technology must be evaluated empirically and rigorously to determine any significant and quantifiable improvement (Pfleeger et al., 1997; Pfleeger et al., 2000). Although a single evaluation can never embrace all possible situations, it is reasonable to seek some evidence of a technology's likely efficacy when used under certain conditions (Fenton et al., 1994).

1.2. Outline of the Solution

It is believed that by combining semi-formal and formal notations, many software engineering problems could be solved. A formal notation ensures system correctness, but it is inaccessible to many practitioners. Despite being less formal, a semi-formal notation is approachable and supports refinement activities. Harmonising semi-formal and formal notations may overcome one notation's limitations while enhancing the strengths of both. Moreover, a formal notation normally appears as textual whereas a semi-formal notation is mainly graphical. By integrating formal and semi-formal notations, the visualisation of graphical representation can be combined with the expressiveness of textual representation.

There are many instances of semi-formal and formal notations integration. One such integration is to combine the formal notation used in B and the semi-formal notation of UML. The rationale behind this integration is that B has strong industrial supporting tools such as Atelier-B (ClearSy, 2003) and B-Toolkit (B-Core, 1999), and the UML has become the de facto standard for system development (Pender, 2003). Supportive environment of UML and B has encouraged the research community to devote significant effort in establishing links between them since a decade ago (Shore et al., 1996; Sekerinski et al., 1998; Meyer et al., 1999; Ledang et al., 2002; Lano et al, 2004; Idani et al., 2006; Snook et al., 2006).

Inventors often assert that their technologies are capable of increasing productivity, delivering better quality product, lowering development cost and enhancing user satisfaction. However, it is no longer acceptable in software engineering to claim a technology as being able to bring benefits without providing scientific evidence of its application. Empirical evaluation helps to determine the effectiveness of a proposed technology (Zelkowitz et al., 1998) where it provides an excellent mechanism to learn what works and what does not work (Basili et al, 1986). Besides, it can eliminate the influence of assumptions and serve for exploring explanations for new phenomena in order to develop new theories (Tichy, 1998; Pfleeger, 1995; Perry et al., 2000). Inventors are not the

best people to perform objective, rigorous evaluations of their own technologies (Kitchenham et al., 2002a; Jeffery et al., 2002). Hence, independent assessors should conduct such evaluations.

Effectiveness of a technology is normally gauged by its ability to achieve certain product or process qualities. Usability has been recognised as an important product quality (McCall et al, 1977; Boehm et al., 1978; Grady et al., 1987; IEEE, 1990; ISO 9126-1, 2001), which determines whether a technology is efficient, effective and satisfying for those who use it in the intended contexts. This research investigates the usability of a development method that combines the formal notation of B and the semi-formal notation of UML, namely UML-B (Snook et al., 2006). As a new technology, an independent usability assessment on UML-B is highly desired. Only if usable, the technology would be more likely adopted by practitioners.

Empirical methods for evaluating a technology can be in the form of controlled experiments, case studies and surveys (Fenton et al., 1996). Although very useful, these research methods are flawed differently (McGrath, 1995). Instead of tackling a research problem with a single method, it has been suggested to perform several individual methods where each examines different but complementary aspects (Daly, 1996; Wood et al., 1999; Perry et al., 2000). In fact, a viable investigation uses multiple methods, which are chosen in such a way that each method's limitations are complemented by the strengths of others (Creswell, 2002). Multiple methods enable the use of two different approaches of empirical assessments, namely quantitative and qualitative. This research comprises a series of controlled experiments and surveys. The controlled experiments assessed the comprehensibility of a UML-B model from the viewpoint of stakeholders for software validation and maintenance purposes. Since it is necessary to consider the usability of UML-B from the viewpoint of developers during the creation of a UML-B model, a survey was designed for that purpose. The worthiness of methods such as UML-B depends on the utility of verification tools. A set of surveys was thus designed to evaluate the usability of the available verification tools such as ProB (Leuschel et al., 2003) and B-Toolkit. The controlled experiments mainly employed the quantitative approach whereas the surveys were

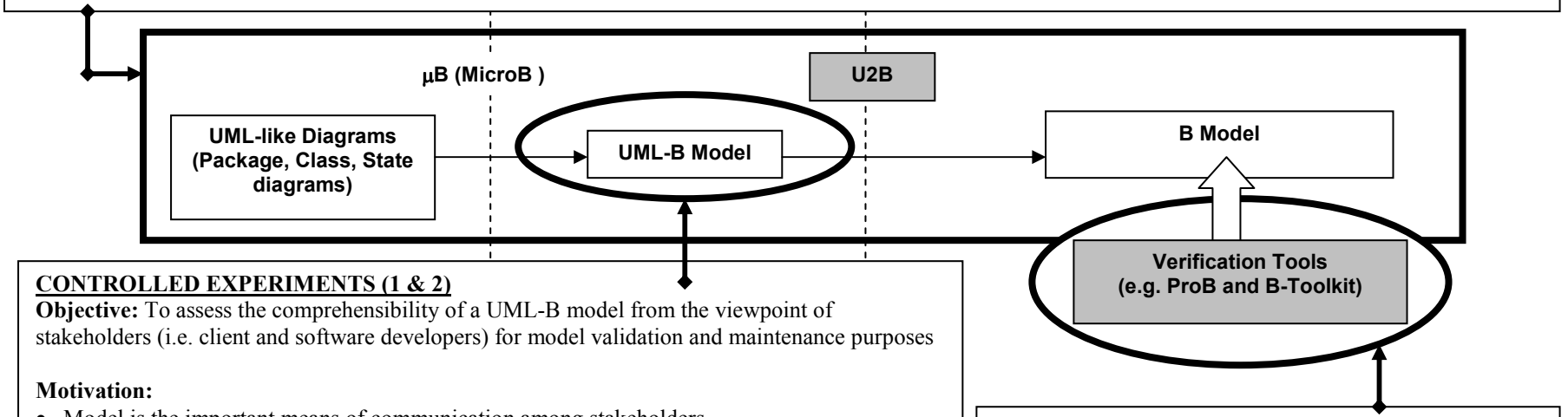
qualitative in nature. However, the controlled experiments and surveys also included some qualitative and quantitative aspect respectively.

Drawing general conclusions from empirical studies in software engineering is difficult as the results depend on a potentially large number of relevant context variables (Basili et al., 1999). The findings of one study are insufficient to provide the essential evidence for confirming the phenomenon. Replications are thus necessary to build up knowledge whether the results hold under different conditions. This research contains replications of the main studies. In particular, there were two controlled experiments conducted on the UML-B model while two surveys on the UML-B method. The investigation was evolutionary where the results of the former studies were confirmed and refined in the latter studies. Overall, there are five empirical assessments conducted in this research as shown in the Figure 1.1 below. The figure is intended to depict the focus of the assessments on the respective aspects of UML-B. The detailed description of UML-B and the related notations and methodologies such as UML, B and Event-B will be included in the next chapter.

SURVEYS (1 & 2)
Objective: To assess the usability of the UML-B method (and its supporting tools) from the viewpoint of software developers for modelling purposes

Motivation: Usability of a technology is essential for its adoption by practitioners in industry

Study and Chapter:
- 1st attempt: **Chapter 4** – Measuring the Usability of the UML-B method
- Replication: **Chapter 6** – Measuring the Usability of the UML-B method – A Survey Replication



CONTROLLED EXPERIMENTS (1 & 2)
Objective: To assess the comprehensibility of a UML-B model from the viewpoint of stakeholders (i.e. client and software developers) for model validation and maintenance purposes

Motivation:
- Model is the important means of communication among stakeholders
- Model comprehensibility is the key foundation to effective validation and maintenance
- Model comprehensibility affects the willingness of practitioners to adopt a technology and user involvement in the validation process

Study and Chapter:
- 1st attempt: **Chapter 3** – Measuring the Comprehensibility of a UML-B model and a B model
- Replication: **Chapter 5** – Measuring the Comprehensibility of a UML-B model and an Event-B model

SURVEY 3
Objective: To assess the usability of verification tools (i.e. B tools)

Motivation: Usability of verification tools is important for methods such as UML-B to be worthwhile

Study and Chapter:
Chapter 7 – Measuring the Usability of Verification tools

FIGURE 1.1: Research Overview

1.3. A list of Original Contributions

The research is the first and the only comprehensive, independent assessment of the combination of semi-formal and formal approaches for software development. The original contributions made by this research are listed below. They have been divided into primary and secondary contributions.

- **(PRIMARY) Empirical investigation into the comprehensibility of models that integrate semi-formal and formal notations**

  The research is the only empirical evaluation that assesses the comprehensibility of graphical formal models as compared to textual formal models from the perspective of software stakeholders. A series of controlled experiments has been conducted (**Chapter 3** and **Chapter 5**). The investigation has provided some interesting evidence of such models' accessibility. In particular, it has shed a light on how future formal methods should be designed for their models to be more approachable to stakeholders.

- **(PRIMARY) Empirical investigation into the usability of methods that integrate semi-formal and formal approaches**

  The research is the only empirical investigation that explores the usability of a development method, which combines semi-formal and formal approaches, from the perspective of software developers. A series of surveys has been conducted (**Chapter 4** and **Chapter 6**). The investigation has provided some evidence, which have been used to generate a set of tentative usability theories of methods such as UML-B. A design profile has also been proposed, which provides future designers some guidelines for designing usable methods.

- **(PRIMARY) Empirical investigation into the usability verification tools**

  The research is the only empirical investigation that attempts to assess the usability of verification tools, which support a method that uses formal notation. A set of surveys has been conducted on two instances of verification tools (**Chapter 7**). The investigation has captured a number of features that are believed to be important for verification tools to become usable. A tentative design guideline for ensuring usable verification tools has been proposed.

- **(PRIMARY) Theoretical usability evaluation of methods that integrate semi-formal and formal approaches**

  The research provides practitioners with some understanding of the strengths, weaknesses, opportunities and threats of methods that integrate semi-formal and formal approaches such as UML-B. A theoretical evaluation that explains *what*, *why*, *who*, *when*, *where* and *how* such methods can be usable has been conducted (**Chapter 8**), based on the findings of the empirical assessments and several theories from Cognitive Science and Educational Psychology.

- **(SECONDARY) Multi-method and multi-discipline approaches to empirical software engineering research**

  The research demonstrates the importance of conducting research using multiple methods where data collected from one method can be used to complement the other. The multiple methods used in the research were controlled experiments and surveys. As one research method focuses on one aspect of interest, combining two methods provides a richer understanding of the phenomenon under study.

The research demonstrates the feasibility of conducting software engineering research using approaches from other disciplines. Software engineering not only concerns technology but also human factors. Therefore, several approaches used in disciplines involving humans were adopted. For instance, the controlled experiments employed one approach from Clinical Trials for the design and analysis (**Cross-Over Trials**). The underlying theoretical background that supports the investigation was adopted from Cognitive Science (**Dual Coding**, **Working Memory**, **Cognitive Load**) and Educational Psychology (**Multimedia Learning**). The surveys used a framework that incorporates Psychology and Human Computer Interaction (**Cognitive Dimensions**) aspects as the instrument. The surveys also adopted an approach for dealing with qualitative data from Social Science (**Grounded Theory**) for the data analysis.

The research combines both confirmatory and explanatory work. The confirmatory work tested a set of predefined hypotheses while the explanatory work endeavoured to discover new and unforeseen insight. The confirmatory work was in the form of controlled experiments (**Chapter 3** and **Chapter 5**) whereas the explanatory work was conducted as surveys (**Chapter 4** and **Chapter 6**, **Chapter 7**). Moreover, both confirmatory and explanatory work employed quantitative and qualitative approaches. The research also demonstrates the importance of confirming empirical results through replication (**Chapter 5** and **Chapter 6**).

1.4. Outline of the Thesis

The above sections have outlined the problem addressed by the research, the solution and the main contributions of the research. The rest of the chapters are organised as follows:

- **Chapter 2: Background Literature**

  This chapter presents the underlying theoretical background that supports the research. It begins with some issues concerning the necessity of evaluating software technologies, and explains several methods and strategies used in empirical research. The chapter also explains the methods and notations involved namely B, UML and UML-B. In addition, a brief description of "Usability" term is also included. Since the usability evaluation in the research concerns the investigation of cognitive processes involved in learning and understanding, several cognitive theories are also discussed.

- **Chapter 3: Measuring the Comprehensibility of a UML-B Model and a B Model**

  This chapter presents the first controlled experiment conducted on a UML-B model. The experiment assessed the comprehensibility of the model in terms of the notation used. The evaluation was based on the comparison made between the notation used in UML-B and the formal notation used in B.

- **Chapter 4: Measuring the Usability of the UML-B Method**

  This chapter presents the first survey conducted on UML-B. The survey assessed the understandability, learnability, operability and attractiveness of the method in supporting model creation task.

- **Chapter 5: Measuring the Comprehensibility of a UML-B Model and an Event-B Model**

  This chapter presents the second controlled experiment conducted on a UML-B model. Similar to the first one, the experiment assessed the

comprehensibility of the model in terms of the notation used. UML-B has introduced some new changes to its environment. Therefore, this experiment replicated the first experiment described in **Chapter 3** on the new version of UML-B. In this experiment, a model developed using the new version of UML-B was compared with an equivalent Event-B model. Event-B on the other hand is a formal notation evolved from the classical B.

- **Chapter 6: Measuring the Usability of the UML-B Method – A Survey Replication**

  This chapter presents the second survey conducted on UML-B. This survey replicated the first survey described in **Chapter 4** on the new version of UML-B. Similar to the first one, the survey assessed the understandability, learnability, operability and attractiveness of the method.

- **Chapter 7: Measuring the Usability of Verification Tools**

  This chapter presents a survey conducted on verification tools that support UML-B. The survey explored a set of features that are important and thus must be present in verification tools for them to become usable.

- **Chapter 8: Evaluation and Recommendation**

  This chapter presents a theoretical evaluation that explains *what*, *why*, *who*, *when*, *where* and *how* methods such as UML-B can be usable.

- **Chapter 9: Conclusions and Future Work**

  This chapter presents the conclusions and areas of future work.

Chapter 2

Background Literature

The previous chapter has introduced the problem addressed by the research and outlined the solution. This chapter aims to provide the underlying theoretical background of the research. This research is mainly empirical in nature. Therefore, the discussion begins with some explanation of methods and strategies used in empirical research. As the research concerns formal and semi-formal methods and notations, the terms are defined and several weaknesses of the notations are discussed. This is followed by specific examples of formal and semi-formal methods and notations that are related to this research, namely B and UML. Later, the chapter explores the idea of integrating UML and B as a possible way of overcoming the problems of formal and semi-formal notations discussed earlier. A specific instance of such integration, which is also the object of study, namely UML-B is also included. The quality aspect focused in this research is usability, thus a brief description of usability is presented. Since usability assessments involve human cognitive activities such as understanding, several cognitive theories are also discussed. An overview of empirical studies on similar aspects is included at the end of the chapter.

2.1. Empirical Research

Empirical research is a study that compares "what we believe" to "what we observe" (Perry et al., 2000). Empirical research as conducted in the field of software engineering aims at providing a scientific and thus more rational basis for evaluating, predicting, understanding, controlling and improving the tools, methods and techniques used in software engineering (Basili et al., 1986). The experimentation of methods, tools and techniques is important because software engineering community needs to improve its knowledge on how software is developed, the effects of various technologies and the areas that most need improvement (Basili et al, 1988; 1996). Besides, empirical research can eliminate the influence of assumptions and alternative explanations, and serves for exploring and finding explanations for new phenomena in order to develop and support new theories (Tichy, 1998; Pfleeger, 1995; Perry et al., 2000).

Empirical research comprises qualitative and quantitative approaches. Qualitative approach is the non-numerical examination and interpretation of observations for the purpose of discovering underlying meanings and patterns of relationships. It aims to examine objects in their natural setting and interpret a phenomenon in terms of explanation that people bring to them (Miles, 1994). In contrast, quantitative approach is the numerical representation and manipulation of observations for the purpose of explaining a phenomenon that those observations reflect. It aims to get a numerical relationship between several variables or alternatives under examination (Juristo et al., 2001). In software engineering, the blend of technical and human behavioural aspects lends itself to combining qualitative and quantitative approaches (Seaman, 1999). These two approaches should be considered as complementary rather than competitive. To understand the impact of human factors on the efficacy of various technologies, quantitative results must be complemented with qualitative analysis and an investigation of subjective, human perceptions (Briand, 2007).

Empirical research can be exploratory or confirmatory, depending on the goal of the investigation. Exploratory study aims to discover new and unforeseen insight

whereas confirmatory study begins with some type of hypothesis or proposition and aims to confirm it (Seaman, 2007). To conduct a confirmatory study, researcher has to know what and how to measure and be able to define a problem by means of hypothesis and its associated variables. Otherwise, an exploratory study is more appropriate. Having said however, a confirmatory study does not have to be always quantitative. Similarly, an exploratory study is not necessarily qualitative in nature.

Empirical research has a number of different methods (Easterbrook et al., 2008). The most common ones include surveys, case studies and controlled experiments. Surveys are used to gather information from a broad population of individuals to describe, compare and explain knowledge, attitudes and behaviour (Kitchenham et al., 2002b). In contrast, case studies investigate a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident (Yin, 2003). On the other hand, controlled experiments investigate a testable hypothesis where one or more independent variables are manipulated to measure their effect on one or more dependent variables (Easterbrook et al., 2008). In general, the key differences between these methods are level of control, research environment, investigation cost and ease of replication. The detailed elaboration on the differences can be found in the literature (Fenton et al., 1996, Wohlin et al., 2000; 2003; Easterbrook et al., 2008).

Empirical research may combine various methods and approaches when studying a phenomenon. The strategy is called mixed methods (Creswell, 2002), multi-method (Brewer et al., 1989) or triangulation (Martin, 1982), which emerged in the recognition that each method has limitations that can be compensated by the strengths of other methods. Rather than using a single method that may be insufficient to explain the phenomenon under study, several individual methods where each examines different but complementary aspects should be used (Daly, 1996; Wood et al., 1999; Perry et al., 2000).

Multi-method empirical research contains two or more studies that employ two or more different methods. For example, it is possible to have controlled experiments and surveys together to investigate a phenomenon. One study complements

another study by helping to confirm the findings, generate research hypotheses or explain the findings. The strategy also includes combining qualitative and quantitative approaches. The main consideration when using the strategy is the sequence in which different approaches and methods are employed. One quantitative study may be followed by a qualitative study or vice versa to assist in explaining and interpreting the findings of the former study. Besides, several studies with different methods may be conducted concurrently to confirm, cross-validate or corroborate findings (Creswell, 2002). Several studies that investigate the same phenomenon using different methods enable more confidence to be placed on the findings. In addition, they also provide various types of evidence. Evidence obtained from different methods and approaches is very useful for improving the understanding of a phenomenon and achieving a cohesive body of knowledge.

Drawing conclusions from a single study is very risky due to its low confirmatory power, a large number of potential factors that interact with the treatment and the challenges of experimental validity. Multi-method is one strategy to alleviate the problem. Another way is by replication where a study is repeated through a series of similar studies. Replication may or may not vary the hypotheses of the main study, and may extend the theory (Basili et al., 1999). Replication enables researchers to gather enough evidence from a number of related studies before a definite conclusion can be made about a phenomenon.

Much of the existing technology has been adopted on the basis of expert opinion and anecdotal evidence rather than on the basis of empirical evidence (Rainer et al., 2005; Kitchenham et al., 2007). This is mainly due to lack of empirical validation in the published computer science and software engineering articles (Lukowicz et al., 1995; Zelkowitz et al., 1997). The situation is however improving as the percentage of articles with no empirical validation has dropped in major publications (Zannier et al., 2006; Zelkowitz, 2007). Among the conducted empirical validations, controlled experiments (Sjøberg et al., 2005) and surveys (Höfer et al., 2007) constitute only a small fraction of the published articles. Lack of conducting controlled experiments in software engineering has been regarded as one reason of software engineering immaturity (Ebert, 1997).

This is because only through experiments, practitioners could gain more understanding of what makes software good and how to make software better (Pfleeger, 1999). It seems that more controlled experiments and surveys should be conducted to understand various aspects of software product and process.

This research adopts the multi-method strategy. It comprises two studies using two types of research methods, namely controlled experiments and surveys. The controlled experiments were mainly confirmatory as they aimed to confirm a set of predefined hypothesis about the phenomenon under study (**Chapter 3** and **5**). The surveys were exploratory as little was known about the phenomenon and thus aimed to discover more of its characteristics (**Chapter 4**, **6** and **7**). The studies complemented each other where they helped to confirm the findings and explain the phenomenon from two different perspectives. There were replications where one study was repeated twice under different conditions (**Chapter 5** and **6**). Despite being confirmatory or explanatory, each study (except the survey described in **Chapter 7**) adopted both qualitative and quantitative approaches to some degree.

2.2. Formal and Semi-formal Methods

Formal methods are defined as methods that impose the use of mathematically based approaches to software development. Several examples of formal methods include VDM (Jones, 1990; Fitzgerald et al., 1998), Z (Spivey, 1992, Bowen, 1996) and B (Abrial, 1996). The rigour of formal methods could improve software development process and produce software with better structure, greater maintainability and fewer errors (Hinchey, 2002). Formal methods are seen as a fault avoidance technique that aims to reduce the introduction of errors into a system. The methods are therefore employed at the early stages of system development, particularly from the specification stage.

The development of a system specification using a formal method allows practitioners to work at an abstraction level that is independent of the implementation (Plat et al., 1992; Harry, 1997; NASA, 1998). Nevertheless, a formal specification cannot be specified using natural languages. Rather, it must be written in a formal notation that is based on a rigorous mathematical model or a standardised programming specification language (IEEE, 1987; 1998). Specifically, a specification is formal if it expressed in a notation that has three components (van Lamsweerde, 2000): rules for determining the grammatical well-formedness of sentences (syntax); rules for interpreting sentences in a precise, meaningful way within the domain considered (semantics); and rules for inferring useful information (proof theory), which provides the basis for automated analysis of the specification. A formal specification is thus an unambiguous and precise description of system requirements that can be rigorously validated and verified. Without a formal specification, a reliable synthesis and analysis prior to an implementation becomes difficult if not impossible (Alexander, 1996).

Even though many do accept the existence of formal methods and their applications in software engineering (Lau et al., 2005), the industry in general remains unconvinced of its utility. One major concern with formal methods is the inaccessibility of formal notation (Finney et al., 1996a; Finney, 1996b; Carew et al, 2005). It is always recommended that formal method specialists should support

practitioners to ensure that the notation is interpreted and employed correctly (Hinchey, 2002; Bowen et al., 1995; 2006). This seems to suggest that practitioners are unable to work independently and organisations have to incur extra cost for hiring experts. The notation is also seen as more usable for programmers rather than for stakeholders who need to specify and validate a specification (van Lamsweerde, 2000).

Graphical notations such as the Unified Modelling Language (UML) (OMG, 2006) are considered as semi-formal because, although they impose some formality and support refinement activities, they cannot be verified systematically to ensure a specification's accuracy and consistency. On the other hand, the visualisation provided by such notations enables system requirements to be specified more naturally and thus promotes better understanding. This is because the symbols used in graphical notations are mainly analogous to the world's objects (Bauer et al., 1993; Stenning et al., 1995). Previous studies have shown that graphical notations have significant advantages for certain tasks. For example, flowcharts have beneficial effects in terms of the time needed to comprehend an algorithm and the accuracy in user response as they illuminate the control-flow of conditional logic (Vessey et al., 1986; Cunniff et al., 1987; Scanlan, 1989). A study has also investigated the use of flowcharts as a supplementary notation to textual code, which revealed some positive evidence in improving comprehension (Curtis et al., 1989).

Despite the claims that graphical notations are somehow better at capturing many aspects of the world naturally, researchers have yet to agree the superiority of graphical notations over textual (Petre, 1995; Blackwell et al., 2001). One reason is that the underlying factors that contribute to the superiority of graphical notations are not well understood (Scaife et al., 1996). Furthermore, textual notations have always been regarded as more expressive. Perhaps it is better to combine graphical and textual notations together rather than using a single notation. This view is similar to integrating semi-formal and formal notations, as the former is mainly graphical whereas the latter is textual. By integrating semi-formal and formal notations, practitioners could benefit from graphical and textual representation and deal with a more accessible formal notation.

2.3. B Method

The B method (Abrial, 1996) is a collection of mathematically based techniques for the specification, design and implementation of software components. The method synthesises many approaches to formal methods such as Z notation (Spivey, 1992, Bowen, 1996), which was also inspired by Abrial, and Stepwise Refinement in Programming (Gries, 1981). In essence, the invention of the B method was motivated by the need to support all stages of software development lifecycle in a uniform and formal way (Schneider, 2001). Unlike most formal methods, this methodological view allows the B method to be applicable beyond only the specification stage. Several organisations have gained the benefits of the B method such as Paris Metro (Behm et al., 1999), Gemplus (Casset, 2002), Clearsy (Pouzancre, 2003) and KeesDa (Hallerstede, 2003). These organisations use the B method as the primary development method for specification and implementation stage.

The B method provides techniques that ensure the consistency of a specification and guarantee the implementation with respect to that specification. In the method, system components are modelled as a collection of interrelated abstract machines. An abstract machine is a specification of what a component should provide. It consists of input, output and a set of allowable operations used to turn the input into the output, which are described using the Abstract Machine Notation (AMN). AMN is a state-based formal specification language which is similar to the Vienna Development Method (VDM) (Jones, 1990). It acts as a uniform notation for describing system states and behaviours at various levels of development, from the specification through to the implementation.

The B method supports hierarchical stepwise refinement where an abstract specification is formally specified and successively transformed into an implementable specification through a number of correctness preserving steps. The abstraction is firstly developed from an informal specification to capture the most essential properties of a system. The abstraction is further refined and decomposed until the system's essential properties are fully specified. After each

refinement, more concrete properties of the system are obtained. Once the properties have been specified formally, further refinement and decomposition allow implementation decisions to be added. A refinement that is at a sufficiently low level can be translated automatically into code.

The refinement of system components from the specification to implementation requires the execution of several proof activities. In particular, the method enforces the discharge of proof obligations, which are the properties that must hold for the components to be self-consistent. There are two main proof activities involved in the B method, which include *Consistency Checking* and *Refinement Checking*. *Consistency Checking* ensures that the component preserves its state conditions whereas *Refinement Checking* ensures that the component is valid at each refinement level. Several industrial tools support the proof activities, namely Atelier-B (ClearSy, 2003) and B-Toolkit (B-Core, 1999). The tools generate proof obligations and prove the obligations through automatic and interactive provers. While the automatic prover discharges the proof obligations automatically, the interactive prover requires user intervention for the proof activities to complete. The automatic prover is normally capable of proving majority of proof obligations. However, some complex proof obligations need to be proved interactively by users through the interactive prover. Discharging proof obligations with the interactive prover may be complicated, but it provides users with a better insight into the system properties and behaviours.

Besides the industrial tools, there are also tools developed within the research community. ProB (Leuschel et al., 2003) for instance, supports the automated *Consistency* and *Refinement Checking* (Leuschel et al., 2005; 2006) via Model Checking (Clarke et al, 1999). Unlike other B tools, ProB comprises a model checker that explores exhaustively the finite behaviour of a component, an animator that executes the operations and a graphical tool that displays the states and transitions covered by the model checker. The tool performs the model checking by verifying a component against the specified properties. It traverses all the reachable states of the component, explores the possible states and finds potential problems. ProB discovers inconsistencies through *Temporal Model Checking* and *State-based Model Checking* (Butler et al, 2005). Through

Temporal Model Checking, the tool attempts to find a sequence of operations that leads to a problem from a valid initial state. On the other hand, the *State-based Model Checking* or *Constraint-based Model Checking* points directly to a valid state that could lead to a problem when a certain operation is invoked. ProB aims to support the interactive proof activity in the industrial B tools. It eliminates some non-trivial errors before more complicated proof activities are performed using the B tools (Leuschel et al., 2006). Its animation that allows the simulated behaviour of a model to be observed provides a useful mechanism for performing validation. Users are provided with the description of the current state, the history that led to the current state, and the enabled operations along with proper argument instantiations. ProB is regarded as a faster and cheaper tool than provers while being more rigorous than manual verification techniques such as reviewing (Snook et al., 2004a).

Event-B[1] (Abrial et al., 1998; 2007) is a modelling notation and method evolved from the B method. It is intended for formal development of discrete systems. It uses the ideas of Action Systems (Back, 1990) that emphasises the incorporation of an event perspective. Events in Event-B replace the idea of operations in B where they can be executed randomly whenever the condition is true. Event-B has been designed with tool support in mind and therefore is surrounded by a set of associated tools for formal verification, model-checking and animation (Abrial et al., 2006). The tools are implemented as plug-ins on the Eclipse platform (Eclipse, 2007). Its notation has two basic constructs, namely *Contexts* and *Machines*. The former contains the static part of a model whereas the latter contains the dynamic parts. Unlike the B method, Event-B does not have a fixed syntax and new constructs can be added whenever necessary (Hallerstede, 2006).

This research includes both variations of B. In the first controlled experiment (**Chapter 3**), the object of study was B whereas in the second one, it was Event-B (**Chapter 5**).

[1] This work is part of the EU funded research project: IST 511599 RODIN (Rigorous Open Development Environment for Complex Systems).

2.4. Unified Modelling Language (UML)

The Unified Modelling Language (UML) (OMG, 2006) has become the de facto standard for system development and is promoted as a technology that helps in producing understandable software systems (Pender, 2003). It is a visual modelling language that is composed of graphical representation to express object-oriented system designs (Fowler, 2004). It emerged as a composition of three object-oriented methods, namely the Booch Method (Booch, 1994), the Object Modeling Technique (OMT) (Rumbaugh et al., 1991) and the Objectory Method (Jacobson et al., 1992) by three primary authors who are referred to as "The Three Amigos". The language has evolved rapidly under the management of Object Management Group (OMG). In fact recently, the organisation has published the latest version of UML 2.0 (OMG, 2006). The version contains new features that enhance UML's capability to represent behavioural and architectural models, business process and rules, and different parts of computing and non-computing disciplines.

UML has been described as a graphical language for visualising, specifying, constructing and documenting the artefacts of a software intensive system (Booch et al., 1999). The language comprises a number of representations such as *Use Case*, *Class* and *Statechart* diagrams for allowing the structural and dynamic aspects of a system to be illustrated from different perspectives. Various ways of representing a system enable UML to be applicable to a variety of modelling domains (Booch, 2002). By providing the relevant views, UML is regarded as a useful means of communication among stakeholders during the development and deployment process (Schmuller, 1999).

Although useful, the graphical aspect of UML has a limitation. In particular, system specific properties and constraints cannot be illuminated using merely the diagrams (Cook et al., 2001; Warmer et al., 2003). Graphical representation in general lacks formality and hence UML models liable to ambiguity and inconsistency. An add-on feature called the Object Constraint Language (OCL) (Warmer et al., 2003) was proposed to bring precision to UML models. OCL is a

precise textual notation, which provides UML models with constraint and object query expressions that cannot be specified by the diagrams. OCL enables the specification of system properties in a more precise and detailed manner than natural language or diagrams alone (Gogolla et al., 2001). OCL is currently a part of the UML standard (OMG, 2006). Although OCL has great potential for improving the correctness of UML models (Hennicker et al., 2001), it has been criticised for being cumbersome and awkward to use (Vaziri et al., 1999). In fact, several authors exclude OCL from their proposals of using UML (Gomaa, 2000) and recommend using OCL only if it is necessary (Larman, 2004; Fowler, 2004). Moreover, OCL lacks of systematic tools that could support its application. Although several tools do available (Toval et al, 2003; USE, 2006), they are isolated and immature. The current OCL users therefore verify their UML models using the conventional verification and validation techniques such as reviewing.

2.5. Integrating UML and B

Formal notations such as used in the B method contain textual mathematical constructs. They have the precision, which enables them to be rigorously verified so that a correct and consistent model can be produced. The notations however are generally difficult for untrained users to understand because they have a large numbers of different symbols that represent complex operations (Britton et al., 1999). Even with training, several studies have found that formal notations are not easily readable and understandable (Finney et al., 1996a; Carew et al., 2005).

Semi-formal notations such as UML mainly use graphical symbols to represent structural and dynamic aspects of a system. Graphical symbols may be effective for communicating ideas but they are not as expressive as textual constructs. In particular, one cannot use graphical symbols alone to specify system constraints. As the expressive power of a representation is essential for model completeness (Brun et al., 1995), a model that uses graphical symbols may be incomplete. Moreover, the lack of formality in graphical symbols causes model verification to be difficult, if not impossible.

A study has indicated that the presence of graphical symbols together with a formal textual notation could improve the notation's readability (Zimmerman et al., 2002). This may suggest that by having UML diagrams with the formal notation of B could also lead to similar results. In fact, a preliminary case study on the idea of integrating UML and B has shown that B promotes a model's precision while UML allows the model to be more intuitive (Satpathy et al., 2001). In essence, integrating both notations may address the lack of formality in UML while improving the accessibility of B. Furthermore, such integration could also benefit from the B's strong industrial supporting tools and the wide acceptance of UML in industry.

The research community has attempted to establish links between UML and B (Shore et al., 1996; Sekerinski et al., 1998; Meyer et al., 1999; Ledang et al., 2002; Lano et al, 2004, Snook et al., 2004a). These studies mainly investigate the

ideas of transforming a UML model to a B model. However, there are also studies that investigate the ideas of reverse transformation, that is, from a B model to a UML model (Idani et al., 2006). Regardless of direction, all these studies aim to exploit the strength of B tools while remaining in a standard industrial process based on UML. They are also motivated by the belief that UML could help B notation to be approachable to practitioners.

This research aims to investigate one such integration, which was developed by a team of researchers from the University of Southampton, namely UML-B (Snook et al., 2006). UML-B was originated as a part of a PhD thesis (Snook, 2002). Since then, the method has been further enhanced and developed in several projects funded by the European Commission such as the Methodologies and Technologies for Industrial Strength Systems Engineering (MATISSE, 2002), the Paradigm Unifying System Specification Environment for Electronic Design (PUSSEE, 2003) and the Rigorous Open Development for Complex Systems (RODIN, 2004).

UML-B is a graphical formal modelling notation and method based on UML and B. It uses UML's *Package*, *Class* and *Statechart* diagrams as the graphical representation of its model. The graphical representation is equipped with formally defined semantics by using an integrated and action language called μB or microB. μB is based on the B's AMN notation. A translator called U2B (Snook et al, 2004b) translates a UML-B model to a textual B model so that B tools can be executed to verify the model. The Figure 2.1 below illustrates the transformation of a UML model to a B model in UML-B.

In general, UML-B consists of the following:

- UML features, which include *Package*, *Class* and *Statechart* diagrams
- Specialisation of UML features via stereotypes and tagged values
- Structuring mechanisms (systems, components and modules) based on specialisations of UML packages
- UML-B clauses, which is a set of textual tagged values to define extra modeling features for UML entities

- μB, which is an integrated action and constraint language based on B's AMN
- Well-formedness rules

The stereotype is used to specialise and enrich the meaning of the UML features. It also relates the UML features to the B concepts. The tagged values or UML-B clauses are defined to attach abstraction details, which do not belong to the standard UML features. These clauses are related with clauses used in the B notation. Other clauses that do not have direct B equivalent are also provided so that specific details can be added to the UML-B model.

The strength of UML-B is that it combines graphical symbols and textual constructs for formal modelling. UML-B uses graphical symbols to illustrate the key aspects of a system while preserving precision by incorporating formal semantics within the graphical symbols. It hides the textual mathematical constructs of B beneath the more user-friendly graphical features of UML. Moreover, it provides a mechanism for a UML-B model to be translated to a B model and later to be verified by the available B tools.

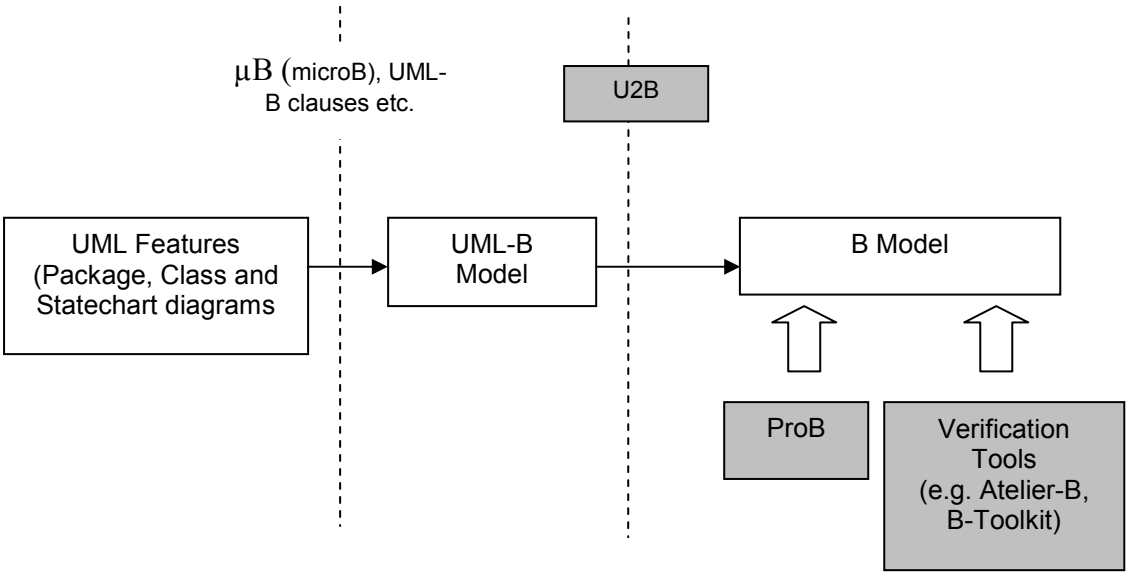


FIGURE 2.1: The transformation of a UML model to a B model in UML-B

During the RODIN project, UML-B has gone through some enhancements. The new UML-B is a graphical formal modelling notation and method based on UML and Event-B (Abrial et al., 1998; 2007). Rather than a specialisation of UML, the new UML-B appears to be a "UML-like" formal modelling language. Although the new UML-B still contains UML features, it is essentially a new notation based on a separate metamodel. The tool support for the new UML-B includes drawing tools and a translator, U2B to generate an Event-B model from a UML-B model. The tools are closely integrated with Event-B tools (Abrial et al., 2006) to enable the automatic generation and verification of Event-B models. Previously, U2B tool resided on Rational Rose (Rational, 2000). The tool has now been redeveloped as an Eclipse (Eclipse, 2007) plug-in to improve its integration with the Event-B tools.

This research assesses both versions of UML-B. In the first controlled experiment, the basic concepts of UML-B were assessed by comparing a UML-B model of the earlier version of UML-B with a B model (**Chapter 3**). Later, a UML-B model developed using the new UML-B was compared with an Event-B model (**Chapter 5**). In addition, the supporting tools of both versions of UML-B were assessed in the surveys (**Chapter 4**, **6** and **7**).

2.6. Usability

The research concerns the use of semi-formal and formal notations in developing a specification or conceptual modelling. There are three elements involved in conceptual modelling: the process of creating the model, the modelling language or notation and the model itself (Piattini et al., 2005). In achieving a high-quality model, the process and the notation used should also be of high quality. The assessment of process quality could be based on the ISO 9000 standard (ISO 9000, 2004) and the Capability Maturity Model Integration® (CMMI) (SEI, 2005). The notation and the model are considered as products and therefore they could adhere to the ISO 9126 standard for product quality (ISO 9126-1, 2001).

There are specific quality criteria defined for the notation used in conceptual modelling such as *domain appropriateness*, *comprehensibility appropriateness*, *executability appropriateness* and *knowledge externalisability appropriateness* (Krogstie, 1998). *Domain appropriateness* is the ability of the notation to capture the problem domain. *Comprehensibility appropriateness* is how easily the modeling notation could be learned, used and understood by users. *Executability appropriateness* is to what extent the notation is formalised to enable execution. *Knowledge externalisability appropriateness* is how relevant knowledge of the problem domain may be articulated in the notation. There are also some other criteria such as the notation's *expressive power*, *generative capabilities*, *extensibility* and *usability* (Brun et al., 1995).

Similarly, there are also various criteria proposed for assessing a model quality. Each of the proposed quality criteria becomes more or less important depending on the task and the stakeholders involved (Wand et al., 2002). There are however certain criteria that are generally agreed by most researchers to be important for both the notation and the model. One of these criteria is that the notation and model should be as easy as possible for users to understand (Batini et al., 1991; Davis et al., 1993; Farbey, 1993; Boman et al., 1997; Moody et al., 1998; Olive, 2000; ISO 9126-1, 2001) so that users are not compelled to put effort into

decoding them (Green, 1980). It seems that understandability or generally usability is the most important quality to measure in conceptual modelling.

Usability has been recognised as an important software quality attribute and thus has been included in major software quality models and standards (McCall et al, 1977; Boehm et al., 1978; Grady et al., 1987; ISO 9126-1, 2001). The primary idea of usability is that a product is designed with the user's psychology and physiology in mind. A usable product is expected to be understandable, efficient to use, easier to learn and satisfying.

There are several standard definitions of usability in the literature, which include:

- The extent to which the product is convenient and practical to use (Boehm et al., 1978)
- The ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component (IEEE Std. 610.12, 1990)
- The capability of the product to be understood, learned, used and liked by the user, when used under specified conditions (ISO 9126-1, 2001)
- The extent to which a product can be used by specified users to achieve specified goals with effectiveness, productivity, satisfaction and safety in a specified context of use (ISO 9126-4, 2004)

Usability is often associated with the functionalities of the product such as defined above and the characteristics of the user interface (Shneiderman, 1980, 1998; Nielsen, 1994). By considering the definition provided by the International Organization for Standardization (ISO), usability is primarily concerned with the use of the product, the user interface and interaction, the process used to develop the product and the capability of an organisation to apply user-centred design (Bevan, 2001). The objective is for the product to be effective, efficient and satisfying when used in the intended contexts. To achieve this, an appropriate interface and interaction has to be designed, which requires a user-centred design process.

Usability in ISO is further defined into five sub-characteristics as follows:

- **Understandability** - the capability of the product to enable the user to understand whether the product is suitable and how it can be used for particular tasks and conditions of use
- **Learnability** - the capability of the product to enable the user to learn its application
- **Operability** - the capability of the product to enable the user to operate and control it
- **Attractiveness** - the capability of the product to be liked by the user
- **Usability Compliance** - the capability of the product to adhere to standards, conventions or regulations

In this research, the usability of UML-B is assessed. This includes the understandability or comprehensibility of the notation used in a UML-B model (**Chapter 3** and **5**), the learnability, operability and attractiveness of UML-B's notation and modelling environment for supporting conceptual modelling (**Chapter 4**, **6** and **7**).

2.7. Cognitive Theories

In the following paragraphs, the two main cognitive theories involved in the empirical assessments are described. The **Comprehension Strategies** was included as one of the qualitative measures for the controlled experiments (**Chapter 3** and **5**) and the **Cognitive Dimensions** was used as the instrument for the surveys (**Chapter 4**, **6** and **7**). Besides these two theories, there are also other cognitive theories used in the research such as Multimedia Learning (Mayer, 2001) and Cognitive Load (Chandler et al, 1991; Sweller, 1999). The description of those theories is included in the respective parts of the thesis to assist explanation, whenever applicable.

2.7.1. Comprehension Strategies

Program comprehension or the understanding of code is identified as a critical cognitive activity in programming (Brooks, 1983; Koenemann et al., 1991). Due to the formality enforced in formal notations, a formal specification and a program seem to be similar in many ways. In particular, they are written using specific predefined symbols and rules that govern how the symbols should be manipulated to represent certain semantics. In fact, several empirical researches have discovered that understanding a formal specification is no more difficult than a program (Snook et al., 2001; Snook et al., 2004c). Perhaps the cognitive activity and complexity involved in program comprehension are also applicable to formal specifications.

There are two key aspects involved in program comprehension, which are direction and breadth of comprehension. The direction of comprehension concerns whether the developer employs *Top-down* comprehension (Brooks, 1983, Soloway et al., 1982), *Bottom-up* comprehension (Schneiderman et al., 1979, Pennington, 1987) or combination of both strategies. The breadth of comprehension concerns the strategies that the developer employs to become familiar with a model, that is, whether through *Systematic* or *As-needed* strategies (Littman et al., 1986).

The *Top-down* comprehension strategy involves the construction of knowledge about the model domain and mapping the knowledge to the model. The process begins by constructing a general hypothesis about the model based on high-level information such as title and model description. The general hypothesis leads to the expectation of certain objects or characteristics in the model, which later generates another level of more specific subsidiary hypotheses. As the model is explored further, the subsidiary hypotheses are refined and evaluated in a depth-first manner. The verification and rejection of the hypotheses depends on the presence of recognisable and familiar features in the model that act as cues to the presence of certain structures. The strategy continues for several successive refinements and verifications of hypotheses until the whole model has been understood.

The *Bottom-up* comprehension strategy involves the encoding and chunking of several individual parts of a model into higher level of abstractions. An individual part is grouped together with other related parts to form a semantic representation of a larger unit. The abstractions are aggregated further until a high-level understanding of the model is attained. This strategy consumes both working and long-term memories. The working memory is used for the encoding of individual parts while the long-term memory is used for the formation of the chunks and understanding of the whole model. The process of comprehension continues by recognising the semantic relationships between the constructed chunks and joining these chunks into higher-level chunks.

While the *Top-down* and *Bottom-up* strategies can be employed for any comprehension purposes, the *Systematic* and *As-needed* strategies are particularly concerned with comprehension for maintenance. The S*ystematic* comprehension strategy is employed to gain a broad understanding of a component. The objective of this strategy is to understand the overall design of a component so that the necessary modifications to be made could fit with the existing design. On the other hand, the *As-needed* strategy is not concerned with the overall design of the component but only the selected local parts that are considered to be relevant. Developers use this strategy to understand the minimum amount of information necessary to successfully carry out the modification. The *Systematic* strategy

acquires both static knowledge and causal knowledge while the *As-needed* strategy acquires only the static knowledge. The static knowledge is the information about the local structure of the component and the causal knowledge is the information about the interactions between various parts in the component.

It is believed that users employ certain strategies in understanding a notation. These strategies if known could lead to the improvement of the notation and the identification of necessary tools and training to support them in the process. In this research, the theories of program comprehension described above were applied to investigate the direction and breadth of comprehension employed by subjects when understanding UML-B and B/Event-B models (**Chapter 3** and **5**).

2.7.2. Cognitive Dimensions

The Cognitive Dimensions of Notations (CD) is a framework that provides a comprehensive vocabulary for discussing the usability of programming languages, tools and environments. It was originally proposed as a broad-brush discussion tool, offering a vocabulary to discuss the usability tradeoffs that occur when designing programming environments (Green, 1989; Green et al., 1996). Nonetheless, it is also applicable beyond the programming environment. It can be employed to a wide variety of notations, both interactive artefacts such as word processors and non-interactive artefacts such as musical notation. Since its proposal, the framework has been used as a basis of usability evaluation for several notations such as UML (Cox, 2000; Kutar et al., 2002), C# programming language (Microsoft, 2007) (Clarke, 2001), spreadsheet application (Tukiainen, 2001), B notation (Snook, 2002), Z notation and tools (Triffitt et al., 2002).

The framework is generally seen as a tool that aids the usability evaluation of information-based artefacts (Green et al., 1998). The framework comprises a number of defined terms, which have been chosen to be easy for non-specialists to comprehend while yet capturing a significant amount of psychology and Human Computer Interaction (HCI) aspects. Since it is intended for non-specialists, it contains general description and checklist rather than detailed analysis. Unlike

many other approaches in HCI, the framework concentrates on the processes and activities rather than the finished product. Furthermore, it focuses on the notational design rather than interactive situations.

The aim of the framework is to provide general guidelines that can be used to evaluate the usability and suitability of an artefact for a particular setting. An artefact is analysed based on a usability profile that contains a set of cognitive dimensions that guides the artefact's evaluation for a particular activity. Different types of user activity are thus best supported by different profiles. The framework distinguishes six main types of user activity (Blackwell et al., 2003): *Incrementation* involves the addition of new elements; *Transcription* involves the conversion of elements, *Modification* concerns the reorganisation of the existing elements, *Exploratory Design* concerns the discovery and creation of new elements; *Searching* involves looking for certain elements; and *Exploratory Understanding* involves the discovery of elements.

The usability evaluation in the framework is accomplished by considering the perspective of end users who use the artefact. This is particularly relevant because only the users who use an artefact understand its strengths and weaknesses for a particular context of use. Since they need to use the artefact to accomplish other important tasks, the users can visualise how they could use the artefact more effectively if certain dimension issues are addressed. Besides the end users, designers can also use the framework to prompt possible improvements in the design of their artefacts.

There are fourteen dimensions in the CD framework as summarised in the Table 2.1 below. To assist non-specialists to conduct a usability evaluation using CD, a CD questionnaire has been developed (Blackwell et al., 2000). The questionnaire is intended to present the dimensions in general terms, applicable to all information artefacts rather than presenting descriptions specialised to a specific system under consideration.

Although the dimensions are conceptually independent, many of the dimensions are pairwise interdependent (Green et al., 1998). This means although any given

pair can be treated as independent, a change in one dimension usually requires a change in some other dimensions. For example, by reducing a notation's *Viscosity* may not affect its *Closeness of Mapping,* but it is likely to affect other dimensions such as increasing the *Abstraction gradient*. The framework considers this situation as a matter of making compromises or trade-offs in artefact designs.

Dimension	Description
Abstraction Gradient	Level of grouping mechanism enforced by the notation
Closeness of Mapping	Mapping between the notation and the problem domain
Consistency	Similar semantics are presented in a similar syntactic manner
Diffuseness	Complexity or verbosity of the notation to express a meaning
Error-proneness	Tendency of the notation to induce mistakes
Hard Mental Operations	Degree of mental processes required for users to understand the notation and to keep track of what is happening
Hidden Dependencies	Relationship between two entities such that one of them is dependent on the other but the dependency is not fully visible
Premature Commitment	Enforcement of decisions prior to information needed and task ordering constraints
Progressive Evaluation	Ability to evaluate own work in progress at any time
Provisionality	Flexibility of the notation for users to play with ideas
Role-expressiveness	Purpose of an entity and how it relates to the whole component is obvious and can be directly implied
Secondary Notation	Ability to use notations other than the official semantics to express extra information or meaning
Viscosity	Degree of effort required to perform a change
Visibility/Juxtaposibility	Ability to view every component simultaneously or view two related components side by side at a time

TABLE 2.1: The Cognitive Dimensions

In essence, the CD provides a framework for assessing the usability of building and modifying information structures. As usability depends on the structure of the notation and the tools provided by the environment, the dimensions are indeed applicable to the whole system. This means the dimensions take into account the notation or the information structures and the tools that support them. In this research, the framework was used to evaluate the usability of the notation used in UML-B and its modelling environment (**Chapter 4** and **6**), and B tools (**Chapter 7**).

2.8. Overview of Related Work

The research investigates the usability of UML-B. The controlled experiments assessed the comprehensibility of the notations used in a UML-B model and a B/Event-B model from the perspective of users who interpret the models (**Chapter 3** and **5**). The surveys investigated the usability of UML-B's notation and modelling environment from the perspective of users who use the method for creating a UML-B model (**Chapter 4** and **6**). In general, there are two main tasks involved in the investigations: model interpretation and model creation.

A number of empirical studies have investigated the efficacy of different representations for model interpretation and model creation tasks. For model interpretation, there are studies that compared representations of different technologies such as Data Flow Diagram (DFD) versus Object-Oriented Technology (OO) (Agarwal et al., 1999), OO versus Extended Entity-Relationship (EER) (Shoval et al., 1994), DFD versus task-oriented menus (Nosek et al., 1986), and Unified Modelling Language (UML) versus Open Modelling Language (OML) (Kim et al., 2000). Several studies compared internal properties or structures of one particular representation such as optional versus mandatory properties of Entity-Relationship Diagram (ERD) (Bodart et al., 2001), different decomposition (Burton-Jones et al., 2002) and types of UML diagrams (Torchiano, 2004). There are also studies that assessed a representation presented using different modes such as OO using narration versus animation (Gemino, 2004). These studies mainly used comprehension as the measure of interest, which was determined by the accuracy of the responses given by users who read the presented models. In contrast, the controlled experiments conducted in this research measured the efficiency of comprehension, which took into account not only the accuracy but also the duration of the task.

In terms of model creation, there are comparisons of representations and methods between different technologies. Some examples include DFD versus OO (Agarwal et al., 1996; Wang, 1996), DFD versus OO versus ERD (Vessey et al., 1994), DFD and Integrated Definition Method (IDEF0) (Yadav et al., 1988), Flowchart

and Program Design Language (PDL) (Ramsey et al., 1993), Relational Data Model (RDM) versus EER (Batra et al., 1990), and Logical Data Structure (LDS) versus Relational Data Structures (Jarvenpaa et al., 1989). The most common measures of interest used in the studies include model correctness, ease of use and ease of learning. Some of these measures were also used in the surveys conducted in this research.

Most of the studies mentioned above assessed the representations and methods that are semi-formal. The representations mainly consist of graphical symbols with some "non-formal" textual notations. The textual notations do not contain mathematical constructs and are generally based on natural language. On the other hand, there are studies that explored the readability of formal notations for model interpretation task. For example, comparisons of formal notations with informal specifications (Carew et al, 2005) and code (Snook et al., 2004c), and comparisons of formal notations using different presentation modes (Zimmerman et al., 2002) and structures (Finney et al., 1996a; 1999).

Studies that investigated the efficacy of representations that combine formal (textual mathematical constructs) and semi-formal (graphical symbols) notations empirically are almost non-existent. Based on the literature search made on three major software engineering digital publications, namely IEEEXplore (IEEE, 2007), ACM Portal (ACM, 2007) and SpringerLink (Springer, 2007), only one study that suits the search criteria was found (Briand et al., 2005). The aim was to acquire empirical studies that explored the combination of semi-formal and formal notations. Thus, specific keywords were used in the search which include "Semi-formal and formal notations", "UML-based formal notation", "Graphical formal notation", "Integration of notations", "Experimentation" and "Empirical studies". Other related keywords were also used such as "Comprehension or Comprehensibility", "Usability" and "UML". The search was based on the published papers up until year 2007.

The study conducted by Briand et al. mentioned above assessed the use of Object Constraint Language (OCL) with UML diagrams. OCL is indeed a kind of formal notation as it has a set of predefined syntax and rules of interpretation, which is

based on first-order logic and set theory. In many ways, the use of UML diagrams with OCL is similar to the use of UML diagrams and B/Event-B. Therefore, Briand et al.'s study is comparable with the study in this research. However, there are some differences between the two studies, as illustrated in the Table 2.2 below. In general, Briand et al. explored the use of UML diagrams with OCL as compared to UML diagrams. In contrast, the study in this research compared UML-B with B/Event-B. The former study assessed the use of semi-formal and formal notations (graphical and textual) with a semi-formal notation (graphical) while the latter study with a formal notation (textual). Both studies indicate that using dual notations together is better than one notation. However, the interpretation of the findings and the impacts that they bring are different. The results of Briand et al.'s study seem to suggest using UML and OCL (semi-formal and formal notations) rather than UML alone (semi-formal notation), particularly when practitioners are properly trained and mentored. On the other hand, the findings of this research indicate that introducing some graphical features of a semi-formal notation into a formal notation improves the formal notation's accessibility. It seems that practitioners are more likely to perform better when the formal notation that they employ contains graphical features rather than being textual exclusively. Further elaboration of these findings can be found in later chapters.

Study	Object of study	Independent Variable	Dependent Variable(s)	Methodology	Results	Strengths and Weaknesses
Briand et al., 2005	UML and OCL (Semi-formal & Formal) versus UML (Semi-formal)	Method (Notation)	• Comprehension • Maintenance • Defect Detection **Focus** Effectiveness **Measures** Percentage (%) of Accuracy (Score)	**Design** • Related within-subject, randomised block design • 2 treatments; 4 sessions; 2 case studies • 2 trials (1st & 2nd experiments) • Trial duration: 4 weeks (1st experiment) => 1 session per week; 8 weeks (2nd experiment) => 1 session per fortnight; 150 minutes per session **Subjects** • 4th year Computer & Software Engineering • 4 groups; 2 blocks; balanced (2nd experiment) • 38 (1st experiment); 84 (2nd experiment) • Training given: no information provided **Instrument** • Comprehension: 20 multiple-choice (90 minutes) • Maintenance: 5-6 open-ended (60 minutes) • Defect Detection: List of defects (150 minutes) • Debriefing questionnaires (to support quantitative results)	**Analysis** 1st experiment: 2-way ANOVA (Factor: Method and Ability) 2nd experiment: 3-way ANOVA (Factor: Method, Ability and System) **Findings** UML and OCL is better than UML only after substantial training (statistically significant at 0.05; modest effect)	**Strengths** • Analysis considered confounding effects • Relatively large samples **Weaknesses** • Convenience sample • Subjects were in contact with each other after each session (at least 1 week gap between trials) • Subjects performed two distinct tasks on the same case study in two subsequent sessions • Students as subjects • Size and complexity (toy problem) • No underlying theories

TABLE 2.2: Similarities and differences between "Semi-formal and formal notations" studies

Study	Object of study	Independent Variable	Dependent Variable(s)	Methodology	Results	Strengths and Weaknesses
Razali, 2008	UML-B (Semi-formal & Formal) versus B/Event-B (Formal)	Notation	• Comprehension (Recognition) • Problem Solving (Understanding) **Focus** Efficiency (Effectiveness and time spent) **Measures** Accuracy over time (Rate of Scoring – marks/min)	**Design** • Related within-subject, randomised block design • 2 treatments; 2 sessions; 2 case studies • 2 trials (1st & 2nd experiments) • Trial duration: within 1 day => 100-120 minutes **Subjects** • 3rd year & Master students of Computer Science and Software Engineering • 2 groups; 3 blocks; balanced (2nd experiment) • 41 (1st experiment); 36 (2nd experiment) • Training given: UML (previous course); 8-9 weeks (B); 1 week (Event-B); 1 week (UML-B) **Instrument** • Comprehension: 3 open-ended questions • Problem solving: 3 open-ended questions • Debriefing questionnaires (to support quantitative results)	**Analysis** Cross-over (Period-effect) **Findings** UML-B is better than B/Event-B (statistically significant at 0.05; large effect)	**Strengths** • Analysis considered period effect • More control imposed on subjects (no interaction) • Instruments were based on cognitive theories involved in understanding graphical and textual representations **Weaknesses** • Convenience sample • Not so large samples • Students as subjects • Size and complexity (toy problem) • Period by treatment interaction/carry-over

TABLE 2.2: Similarities and differences between "Semi-formal and formal notations" studies (continued)

2.9. Conclusion

This chapter has provided some theoretical background of the research. The information acts as the basis for the empirical assessments conducted in this research, which will be elaborated in detail in the following chapters. In particular, **Chapter 3** and **5** present the controlled experiments conducted on a UML-B model whereas **Chapter 4**, **6** and **7** discuss the surveys conducted on UML-B's notation, modelling environment and supporting tools. The assessments aim to provide some empirical evidence of the usability of UML-B from two different perspectives using different assessment strategies.

Chapter 3

Measuring the Comprehensibility of a UML-B model and a B model

Software understandability is a characteristic of software quality, which means ease of understanding of software systems (Boehm et al., 1978). In Boehm's quality model for instance, understandability is considered an important aspect of software maintenance. Software maintenance in general accounts for the largest cost in the software lifecycle (Page-Jones, 1980; Sommerville, 2001), where software understandability or comprehensibility plays a crucial and costly role in the software maintenance process (Pigoski, 1996).

Software maintenance involves some modification to be made on the existing system. It is a norm that the maintainers who are responsible for making the modification were not the engineers who designed and developed the system. In fact, the engineers who developed the system originally might not be longer available in the organisation for the maintainers to refer. Furthermore, the system might have been developed many years ago. The only resources available to the maintainers for performing the modification task are therefore the available software artefacts, which are mainly expected to be easy to understand.

Software specification is a fundamental software artefact as it captures what a system should do. It is the primary point of reference for people who deal with a system. For instance, maintainers scrutinise specifications to understand not only the localised properties of a system that need to be changed but also the context within which the changes should take place. These tasks are not straightforward particularly when the notations used in the specifications are difficult to interpret.

The situation becomes more challenging, as they need to make the necessary modification as accurately as possible in a short period of time. Maintenance in essence costs maintainers time, effort and money. This requires that the maintenance process to be as efficient as possible. Specification comprehensibility is necessary for an efficient maintenance process as it allows maintainers to understand the system properties quickly prior to the modification task.

Besides maintenance, specification comprehensibility is also important for specification validation process. Specification validation is concerned with showing that the specification has actually defined the system that the customer wants. It is a process where a number of stakeholders review and check the specification for anomalies and omissions. The validation is critical because if it is not properly done or inadequate, errors in the specification will propagate to the later stages. No amount of verification could overcome those errors even though the development imposes formality to some extent. For instance, many of the errors found in failed safety-critical systems stem from poor or no specifications, not due to incorrect implementation (MacKenzie, 2001). The lack of stakeholder involvement has been the number one factor for such a failure (Standish, 1998; 2001). Having comprehensible specifications should motivate stakeholders' involvement particularly the domain experts during the validation process.

The notation used in a specification plays a vital role. The comprehension process and the subsequent tasks related to it will be affected if the people involved struggle to decipher the notation rather than concentrating on the specification's contents. It has been known that the use of formal notations in a specification increases its precision, which in turn enables greater consistency and correctness to be obtained (Plat et al., 1992; Craigen et al., 1995; NASA, 1998; Hinchey, 2002). Nevertheless, it has been a concern that the notations can also cause comprehension difficulties (Finney et al., 1996a; Finney, 1996b; Carew et al, 2005). The notations are seen as being difficult to comprehend due to the usage of unfamiliar symbols and underlying rules of interpretation that are not apparent to many practitioners. It is believed that the widespread adoption of formal methods in industry will only happen when their notations are more accessible to a wide range of users.

The usefulness of graphical representation in software specifications has been recognised for some time (Vessey et al., 1986; Cunniff et al., 1987; Scanlan, 1989; Curtis et al., 1989). The representation is perceived as easy to understand quickly as it is easier to visualise the mapping of symbols to the real world objects they represent (Bauer et al., 1993; Stenning et al., 1995). A purely graphical representation however is not as expressive as the textual representation as some aspects of system properties cannot be specified completely using just diagrams (Petre, 1995). As a result, informal specifications written using textual representations such as natural language have been the most common and widely accepted approach to specifying requirements (Zave, 1990). Perhaps combining the graphical and textual representations together in a specification could be a strategy to establish synergy between both representations. A combined graphical representation with supporting textual representation can assist visualisation while still achieving the full expressiveness and precision of a textual representation.

Formal notations normally appear as textual whereas semi-formal notations mainly as graphical. By integrating formal and semi-formal notations, practitioners could indeed benefit from both graphical and textual representations. One of the ideas towards this integration is to combine the formal notation used in a formal method, namely B (Abrial, 1996), and the semi-formal notation used in the Unified Modelling Language (UML) (OMG, 2006). A method called UML-B (Snook et al., 2006) is one such integration. The rationale behind this integration is that B has strong industrial supporting tools such as Atelier-B (ClearSy, 2003) and B-Toolkit (B-Core, 1999), and UML has become the de facto standard for system development (Pender, 2003).

This chapter presents an experiment conducted on the notation used in UML-B. The objective was to explore whether the notation could improve the specification or model comprehensibility. The evaluation was based on the comparison made between the notation used in UML-B and the formal notation used in B. The measurement used in the evaluation focused on the efficiency in understanding both notations and performing the required tasks. In the following paragraphs, Section 3.1 to 3.5 explain the technical aspects of the experiment's preparation and execution. Section 3.6 discusses the results and data analysis. Section 3.7

explains several threats to the validity of the results. Finally, Section 3.8 concludes the chapter with a summary of the main findings and future work.

3.1. Objectives

The main objective of this experiment was to evaluate the comprehensibility of the notation contained in a UML-B model compared to a traditional B model. The treatments of the experiment were therefore a UML-B model and a B model. A UML-B model comprises the semi-formal notation used in UML, namely *Class* and *Statechart* diagrams, and the formal notation used in B, namely B notation. A B model comprises only the B notation.

The experiment was conducted to confirm or refute a theory that suggests the notation used in UML-B has a particular effect on stakeholders, making it better in some way than the notation used in B. This also includes another related theory that suggests the integration of graphical and textual representations is more effective in portraying information than a textual representation alone (Mayer et al. 1996). In essence, a UML-B model comprises graphical and textual representations whereas a B model contains only a textual representation. Stakeholders in the context include software developers who view the models for validation and maintenance (enhancement) purposes. The stakeholders are assumed to be new users with limited hours of formal training on the notations used. Clients are also included in the population only if they possess a reasonable amount of knowledge of software development and the technologies involved.

The experiment attempted to answer the following broad research questions:

Is a UML-B model easier to understand (efficiency in understanding and performing the required tasks) than a B model for stakeholders with limited hours of training?

Is graphical representation (semi-formal notation) in concert with textual representation (formal notation) more effective in portraying information?

The standard statistical inference and hypothesis testing (Fisher, 1956; 1990) was adopted in this experiment. The testing involves the construction of null (H_0) and alternative (H_1) hypotheses. The null hypothesis is the statement being tested, which indicates "no effect" or "no difference" in the true means between populations. In essence, the testing is designed to assess the strength of evidence against the null hypothesis. If the evidence is strong, the alternative hypothesis will be accepted. The alternative hypothesis is therefore the opposite of the null hypothesis, which indicates that there is an effect between populations. The strength of evidence is determined by the significance criterion (α) and the p-value (P). The null hypothesis will be rejected and the alternative hypothesis will be accepted only if the computed p-value is less than the α value. The α value represents the risk of incorrectly rejecting the true null hypothesis, which is normally set as 0.05 within the software engineering field. The p-value is the probability that the null hypothesis is true. The smaller p-value indicates the stronger the evidence against the null hypothesis. By comparing the p-value against $\alpha = 0.05$, it means a result will not be considered as statistically significant where the null hypothesis can be rejected unless there is at least 0.95 probability that the conclusion is correct.

The null hypothesis stated for this experiment was:

Null Hypothesis (H_0): The UML-B model is no more comprehensible than the B model

to be rejected in favour of the alternative hypothesis:

Alternative Hypothesis (H_1): The UML-B model is more comprehensible than the B model

The statement of the null hypothesis above implies that there is no difference between the UML-B model and the B model in terms of comprehensibility. On the other hand, the alternative hypothesis indicates that there is a difference between the UML-B model and the B model in favour of the UML-B model. Although statisticians argue that one should always use a two-sided alternative with no specific direction (Moore et al., 2006), a one-sided alternative was employed for this experiment. This is because UML-B can only be considered as worthwhile if its notation could overcome the current barriers against formal notation such as used in B. In other words, the UML-B model should be better than the B model in terms of notation comprehensibility. After all, this is the theory that the experiment aimed to confirm or refute by providing some empirical evidence.

In the process of understanding a model, it is believed that the subjects employ certain comprehension strategies that assist them to perform the task. Therefore, this experiment also aimed to identify the comprehension strategies used by the subjects in understanding both models. The strategies encompass the direction and breadth of comprehension, which include the top-down (Brooks, 1983, Soloway et al., 1982) and the bottom-up (Schneiderman et al., 1979, Pennington, 1987), the systematic and the as-needed (Littman et al., 1986) strategy respectively. The theoretical aspects of these strategies have been elaborated in **Chapter 2**.

3.2. Design

The experiment had a related within-subject design where each of the subjects was trained and assigned a task on both models. As there were two treatments to be tested in the experiment, the subjects were allocated randomly into two groups; *Group X* and *Group Y*. To reduce variability across groups, the blocking technique

was applied. The subjects were blocked based on their ability on the object-oriented technology and formal methods. The subjects' grades on the respective courses during their studies were used as the basis for the blocking. Each subject from each block was then randomly assigned to one of the groups.

The experiment was designed in such a way that at one point in time, *Group X* was assigned a task on the UML-B model while *Group Y* was assigned the same task on an equivalent B model. The reverse was then carried out later where *Group Y* was assigned a task on the UML-B model while *Group X* was assigned the same task on an equivalent B model, as illustrated in the Table 3.1 below. There was a short break between the two consecutive sessions. The design which is called AB/BA cross-over trial (Senn, 2002) was employed in order to eliminate any task direction bias and subsequently any ability effect. The cross-over trial is a study in which subjects are given sequences of treatments where the object of study is the differences between individual treatments. The cross-over trial is particularly useful for obtaining a number of observations between two treatments when fewer subjects are available. It also helps to remove the problem of large differences between subjects that can obscure treatment effects.

	Group X	**Group Y**
1st session Case 1 (i.e. Auction System)	Tasks on UML-B model	Equivalent tasks on B model
=== BREAK ===		
2nd session Case 2 (i.e. Library System)	Tasks on B model	Equivalent tasks on UML-B model

TABLE 3.1: Group and task allocation

Despite being able to eliminate between subjects variability, there is always a possibility that the cross-over trial could introduce several effects particularly period effect, period by treatment interaction and carry-over effect. A period effect refers to a trend that affects the experiment as a whole. For example, it is possible that there is a general tendency for values in the second period to be higher than those in the first period regardless of the treatment applied. A period by treatment interaction is when the treatment effect differs according to the order

in which treatment occurs. For instance, the treatment effect in the "UML-B then B" sequence is not the same as in the "B then UML-B" sequence. A carry-over effect is where one treatment affects the treatment in a subsequent session. The carry-over effect has its origin in a preceding treatment and is thus order-dependent.

The Table 3.2 below illustrates the basic model for a cross-over trial. The detailed explanation about the model can be found in the literature (Senn, 2002). The model shows the expected responses and the associated statistics for two subjects assigned to two different sequences. The aim is to calculate the difference between the two treatments, UML-B and B. Therefore, the treatment effect (τ) is estimated by adding the cross-over difference of X and Y and dividing it by two, which gives: $\tau - (\lambda_U - \lambda_B)/2$. The division is necessary in order to obtain one treatment effect (τ) instead of twice (2τ). Similar results can also be obtained by subtracting the period difference of Y from X and dividing it by two. A cross-over difference (CD) is the treatment difference between UML-B and B regardless of the sequence. On the other hand, a period difference (PD) is the treatment difference between the first period and the second period. In essence, the CD and the PD of the "UML-B then B" sequence is the same. However, the CD of the "B then UML-B" sequence is minus of its PD (CD= -PD).

The $\tau - (\lambda_U - \lambda_B)/2$ is an unbiased estimate of the treatment effect if $\lambda_U = \lambda_B$ or $\lambda_U - \lambda_B = 0$ or $(\lambda_U - \lambda_B)/2$ is relatively small as compared to the treatment effect (τ). The model demonstrates that the analysis of a cross-over trial eliminates the period effect (π) so that the true treatment effect (τ) can be obtained. For the estimate to be unbiased however, the model assumes that the period by treatment interactions of the two sequences are cancelled out and thus, no period by treatment interaction or that the interaction is smaller than the treatment effect.

The cross-over trial is common in clinical sciences but it is rarely adopted in software engineering field due to complex experimental handling and data analysis. Besides, it is difficult to ensure that there is no period by treatment interactions present in software engineering treatments.

Subject	Sequence	Expected response		Expected value of statistics		
		Period 1 (P1)	**Period 2 (P2)**	**Cross-over difference (UML-B - B)**	**Period difference (P1-P2)**	**Subject Total (P1+P2)**
X	UML-B then B	$\mu_X+\alpha$	$\mu_X+\beta+\pi+\lambda_U$	$(\alpha-\beta)-\pi-\lambda_U = \tau-\pi-\lambda_U$ (i.e. P1-P2)	$(\alpha-\beta)-\pi-\lambda_U = \tau-\pi-\lambda_U$	$2\mu_x+\alpha+\beta+\pi+\lambda_U$
Y	B then UML-B	$\mu_Y+\beta$	$\mu_Y+\alpha+\pi+\lambda_B$	$(\alpha-\beta)+\pi+\lambda_B = \tau+\pi+\lambda_B$ (i.e. P2-P1)	$-(\alpha-\beta)-\pi-\lambda_B = -\tau-\pi-\lambda_B$	$2\mu_Y+\alpha+\beta+\pi+\lambda_B$

Note:
μ_X : the average/mean effect for subject X
μ_Y : the average/mean effect for subject Y
α : the effect of treatment UML-B
β: the effect of treatment B
τ: the difference between treatment UML-B and B (i.e. $\tau = \alpha-\beta$)
π: the period effect (i.e. the expected difference between Period 2 and Period 1)
λ_U: the period by treatment effect due to sequence "UML-B then B"
λ_B: the period by treatment effect due to sequence "B then UML-B"

TABLE 3.2: Expected responses and statistics for two subjects in different sequences of an AB/BA cross-over trial

3.3. Subjects

There were forty-one students that participated in the experiment. This included twenty-seven third-year Undergraduate students and fourteen Masters students of Computer Science and Software Engineering courses at the University of Southampton, United Kingdom. They were students from Europe, Asia and Africa continents. The international students, who came from outside the United Kingdom constituted half of the subjects and the proportion of women to men was 1:4. There were twenty-one students in *Group X* and twenty students in *Group Y*. *Group X* consisted of thirteen Undergraduate students and eight Masters students, whereas *Group Y* had fourteen Undergraduate students and six Masters students.

The subjects were students who registered for the "Critical System" course in Spring 2006 (ECS, 2007). They were taught formally on B for about nine hours and on UML-B for one hour. All subjects had gone through courses on the object-

oriented technology and formal methods at some points of their studies. The subjects therefore were familiar with the notations used in this experiment but were not very experienced. The subjects were aware that the experiment was intended for research purposes. They were initially concerned at their assessment being affected by the experiment. However, they were reassured by the small motivational mark associated with it, which was designed to reflect serious participation in the experiment rather than test performance.

The experiment adhered to the University's ethical policies and guidance for conducting research involving human participants (UoS, 2007). The tasks performed in the experiment were aligned with the expectation of the course and had pedagogical values. The subjects were motivated to participate as the level of understanding tested in the experiment was considered to be necessary for them to do their coursework and prepare for the examination. It served both as revision on B and first practice on UML-B. The qualitative part of the experiment provided a space for reflection on the learning. One of the exam questions was designed to draw on such reflection.

The subjects were in the final semester of their respective courses and had reasonable amount of experience and knowledge of software development. They were the next generation of professionals. Thus, they represented closely the population under study; software stakeholders.

3.4. Variables

The main difference between experiment and any other empirical assessments such as case studies is through the notion of independent and dependent variables. In an experiment, the variables are identified and later the sampling is done over them rather than from them (Fenton et al., 1996). This experiment identified the notations used in the models as its independent variable. The experiment aimed to examine the effect of the notations on model comprehensibility. So, the identified dependent variables were as follows.

3.4.1. Direct and Indirect Measures

The experiment identified two direct measures that acted as its dependent variables:

- **Score (Accuracy):** This variable is the mark obtained. Each question was given a specific allocation of marks. Since the questions were open-ended, the marking was based on specific keywords expected from the answers. Marks were awarded for the presence of these keywords. The questions were carefully constructed so that the marks could be easily decided. Acceptable answers were prepared beforehand. One person did the marking so that there was consistency throughout the process.

- **Time Taken:** This variable is the time taken to answer each question in minutes. The subjects were required to state the *Start time* and *End time* for each question. The *Start time* excluded the time to read and understand the question. The *Time Taken* was determined by subtracting the *Start time* from the *End time***.** It is the time taken by the subjects to understand the model and answer the question.

The *Score* was chosen as the measure of comprehension because the subjects could only answer a question correctly if they understood the object being evaluated. To avoid the formulation of correct answers from wild guess or hunch, the questions were constructed in such a way that the subjects could only derive the answers from the models. The *Time Taken* was decided to be the other measure because it is the most frequent measure of software engineers' effort in any software development, particularly maintenance (Foster, 1991). Moreover, software engineering is not just about developing correct products but developing products in a cost-effective way where the cost is primarily determined by the consumption of development time and budget (Sommerville, 2001). A technology is better than the other if it allows software engineers to do their tasks correctly in least possible time.

The quality aspect measured in this experiment was efficiency in understanding the models. This means a model is considered as more comprehensible than the other if it allows the subjects to answer the questions accurately in a minimum period of time. Therefore, the *Score* and the *Time Taken* measures were used to determine an important indirect measure namely *Rate of Scoring*. The *Rate of Scoring* was obtained by dividing the *Score* by the *Time Taken*. Due to its importance in this experiment, the *Rate of Scoring* was given more attention and emphasis during the data analysis.

The *Rate of Scoring* is believed to be a more meaningful measure of model comprehensibility compared to the *Score* or the *Time Taken* alone because of several reasons. By measuring the *Score*, one could determine the accuracy of the answers. However, high *Score* with shorter time is more desirable than high *Score* with longer time although the level of accuracy is the same. This is because high *Score* with longer time may indicate that the subjects struggled to derive the correct answers. The model comprehensibility would likely have caused this phenomenon, if other confounding factors such as unclear questions had been controlled. Similarly, by measuring the *Time Taken* alone is useless as shorter or longer time does not indicate anything if the accuracy of the answers has not been taken into account. Furthermore, since the experiment was conducted in allocated time slots, most subjects seemed to spend about the same time, whether or not they had answered all the questions.

3.4.2. Other Measures

Empirical assessment comprises both qualitative and quantitative approaches. In software engineering, the blend of technical and human behavioural aspects lends itself to combining qualitative and quantitative approaches in order to take advantage of the strengths of both (Seaman, 1999). In fact, gaining the qualitative measures is important for human-based experiments since more than one interpretation can be placed on the data, which are not readily facilitated by the statistical approaches (Daly, 1996). The qualitative measures in essence can be used to supplement the statistical analysis on the quantitative measures and to explain the individual results. Therefore, the data collection for this experiment

was not limited only to the above quantitative measures. Some other qualitative measures were also gathered to support the quantitative measures, which included the subjective rating of model comprehensibility, the subjects' preference between the model notations and the subjects' personal comments on the models. The debriefing questionnaire, which required the subjects to state how they derived the answer for each question also acted as one of the qualitative measures.

3.5. Materials and Procedure

3.5.1. Design of the Materials

The materials used in the experiment included models written in each notation, a questionnaire on each of the models and a set of debriefing questionnaire on the tasks performed. There was also an instruction sheet that explained the steps required when performing the tasks. Since the experiment had two treatments to be examined in each of the two sessions, four models that represented two separate case studies were developed. In the first session, *Group X* was given a UML-B model and *Group Y* was given an equivalent B model on *Auction System*. In the second session, *Group X* was given a B model and *Group Y* was given an equivalent UML-B model on *Library System*, as illustrated earlier in the Table 3.1. Two separate case studies were needed to avoid learning effects. The models for the second session were made as equivalent as possible to the first session so that the treatment effect to be tested remained the same. However, they were different enough in subject matter to avoid confounding the second session with learning gained from the first session. In each case, the UML-B model had one *Class* diagram with four classes and two *Statechart* diagrams with two states each. On the other hand, there were about 180 lines of script for each of the B models. The models can be found in the **Appendix A**.

3.5.1.1 Questions on Models

The comprehension measure used in this experiment was based on the proposed approach on experimental studies of programmers' behaviour (Brooks, 1980). Several indicators were employed to gauge subjects' comprehension level. Subjects are considered as being able to understand a model if they possess some or all of the following abilities (Finney et al., 1999):

(1) Interpret the symbols used in the notations
(2) Identify right abstractions by their purposes
(3) Understand the relation between inputs and outputs
(4) Understand the mapping between model and domain
(5) Modify by changing the right abstractions
(6) Modify by adding new features to the model

The experiment employed four of the criteria stated above, namely criterion *(1)*, *(3)*, *(4)* and *(6)*. Criterion *(1)* was selected because symbols play an important role in any notation especially in symbol-extensive notations such as employed in formal methods. If subjects cannot interpret the meaning of specific symbols used in a notation, it is almost impossible for them to understand the model. Operations are the heart of any software systems. Subjects should be able to identify the required inputs and trace through the transition steps in the operations to identify the outputs. The relation between the inputs and outputs should be accessible. Therefore, criterion *(3)* was selected to assess this aspect.

Criterion *(4)* was chosen because it is essential for ensuring any model's accuracy and completeness; a quality that is expected from any specification (Davis et al., 1993; Piattini et al., 2005). Stakeholders who are involved in the validation process must understand the mapping between the presented model and the problem domain. Otherwise, the model cannot be checked. Since it is so important, two questions were designed for this criterion. Maintenance involves modification by changing certain system elements and adding new features. Maintainers should be able to execute these activities successfully if they understand the models. Hence, criterion *(6)* was included.

The remaining criteria were not selected due to several reasons. Criterion *(2)* was not included because most abstractions in UML and B are obvious. For instance, clauses in B such as *VARIABLES* and *OPERATIONS* are self-descriptive and thus can be easily interpreted. Similarly, as the subjects were quite familiar with UML, they could easily recognise the role and functionalities of the diagrams and clauses used in the notation. Although there was no specific question on Criterion *(5)*, the question on Criterion *(6)* had indirectly included it. When adding a new feature, it is normal to also change the existing elements to accommodate the new feature.

There were five questions for each model. The questionnaires on both UML-B and B models were similar to each other except for the question on Criterion *(1)*. This cannot be avoided as each notation has its own unique symbols that are important for subjects to interpret in order to comprehend the models. The questions were open-ended in nature rather than multiple-choices. This allowed the subjects to derive the answers independently without being influenced by the given selections. The questions were made simple and straightforward in order to avoid any confusion caused by the words used or the way they were constructed. The clarity and simplicity of the questions were tested in the pilot study, which will be elaborated in the next section. As the objective was to assess the subjects' comprehension level, the questions were constructed using the comprehension keywords proposed in the Bloom's Taxonomy (Bloom et al., 1956). The Table 3.3 below illustrates the mapping between the comprehension ability criteria explained above and the questions.

The subjects were required to read and understand each question before answering. Whenever they were ready to attempt any question, they were needed to state the *Start time*. The time was literally the time shown on the clock displayed in the room or the subjects' own watches. Similarly, they were required to state the *End time* whenever they had provided the answer. They were then instructed to proceed to the respective part of the debriefing questionnaire before continuing with the next question. To ensure the right track was followed, the necessary instructions were included timely. The questionnaire was designed in this way to ensure the recorded time was indeed the time spent for understanding the model and providing the answer. If the subjects were required to state the *Time*

taken instead of *Start time* and *End time*, the situation may have been problematic. For instance, the subjects might have failed to notice the time when they actually started. As a result, they would skip or even give incorrect time spent that included the time to read the question.

Comprehension Ability Criteria	Question
Criterion (*1*): Interpret the symbols used in the notations	**Question A1:** What does the … symbol mean in the model?
Criterion (3): Understand the relation between inputs and outputs	**Question A2:** In your own words (i.e. natural language), describe the difference in output between X and Y operations.
Criterion (*4*): Understand the mapping between model and domain	**Question A3:** By analysing the model, is it possible… …? Which part(s) of the model do(es) indicate you this? **Question A4:** In your own words (i.e. natural language), describe the difference in functionality between A1 and A2 operations. What does the following statement of A1 operation mean exactly in natural language? <*Statement*>
Criterion (*6*): Modify by adding new features to the model	**Question A5:** A new requirement is added to the system: <*Requirement*> How would you introduce/add this requirement to the current model? Formulate your solution explicitly on the given sheets.

TABLE 3.3: The mapping of comprehension ability criteria and questions

3.5.1.2. Debriefing Questions

Besides the questionnaire on the model, the subjects were also provided with a set of debriefing questionnaire. The debriefing questionnaire was the same for both models and sessions. The objective of the questionnaire was to identify the strategies used by the subjects to answer each question, either through *Guess*, *Previous knowledge*, *Common sense*, *Understanding the model* or *Other*. Besides, it was aimed to investigate the direction and breadth of comprehension employed by the subjects when understanding both models. The nature of this questionnaire was multiple selections. The subjects were given a set of answers where they could select one or more selections. Any other answers were permitted if they did

not belong to any of the given selections. The details of the questions used in the experiment can be found in the **Appendix A**.

3.5.2. Pilot Study

The importance of performing a pilot study before the execution of an experiment cannot be over emphasised. Performing a pilot study can mean the difference between a success and a failure of an empirical assessment (Pfleeger, 1995; Kitchenham et al., 2002b). A pilot study was conducted to validate and verify the accuracy of the materials prepared for the experiment. These included the clarity of the instructions, the validity and complexity of the questions and the practicality of the tasks required relative to the time available for the experiment. The pilot study was also intended to identify any issues that might have not been realised during the preparation of the materials.

Five participants who were postgraduate colleagues of the researcher were involved in the pilot study. They were B active users who applied the notation extensively in their respective research. Similar to the subjects, the participants learned UML at some points of their studies. Thus, they were familiar with the notation but were not so experienced. They knew UML-B through reading and seminars but never used it. One of the postgraduates however had some practical exposure to UML-B. They were suitable for the pilot study because they reflected the characteristics of the actual subjects. Their expertise in B was more than the subjects, which may have caused them to overlook some aspects. However, the expertise was seen as necessary to validate the accuracy of the materials especially the B models. The pilot study revealed that some instructions were not clear enough and that some tasks were too complex. As a result, several questions had been discarded and modified. The instructions had been made simpler. These modifications were made in the final version of the materials.

In addition to the pilot study, an expert also reviewed the validity and accuracy of the UML-B models. He was the author of UML-B himself. He had been chosen for the task because he was the only person who knew every detail of UML-B.

Since the review was done together with the researcher who was neutral, no bias had been introduced.

3.5.3. Execution

The experiment was a paper-based exercise, which was conducted in a hundred-minute slot. The slot was divided into two sessions with thirty-five minutes each. In each session, each subject was given a specific model and its questionnaire. The instruction sheet was distributed and explained before the first session began. The materials for the first session were collected after thirty-five minutes had passed and the materials for the second session were distributed right after. During this time, the subjects had a break before starting the second session. After the second session had passed, an additional set of questions was distributed where the subjects were asked about both models comprehensibility subjectively. Five minutes were allocated for answering this qualitative questionnaire.

The subjects were not allowed to communicate with each other or leave the room at any time until the experiment ended. The subjects were separated from each other as if in an examination session. During the tasks however the subjects were allowed to refer to textbooks or notes. The rationale behind this is that stakeholders enquire knowledge about a technology from many possible references. Stakeholders also work in teams where discussion is possible. However, it was not permitted in the experiment to control any external factors. The subjects were also instructed to inform the researcher if they had any trouble in understanding the questions. This was to ensure that any confusion that may have arisen was due to the model comprehensibility rather than the materials.

3.6. Results and Analysis

The experiment employed two types of measures, namely the quantitative and qualitative measures. The quantitative measures involve the comparison of numeric data between models while the qualitative measures mainly involve the subjects' perspectives on the given tasks and models. The measures are treated and elaborated separately as follows.

3.6.1. Quantitative Measures and Analysis

The dependent variables of this experiment were *Score* and *Time Taken*. These direct measures were taken to determine a derived measure namely the *Rate of Scoring*, which was obtained by dividing the *Score* by the *Time Taken*. The *Rate of Scoring* was the measure of interest as it considered the accuracy and the duration of the comprehension tasks. The scale used for the *Rate of Scoring* was *marks per minute* (marks/min). This means a model with a higher *Rate of Scoring* is better than otherwise since it indicates a higher accuracy with least time taken to understand the model.

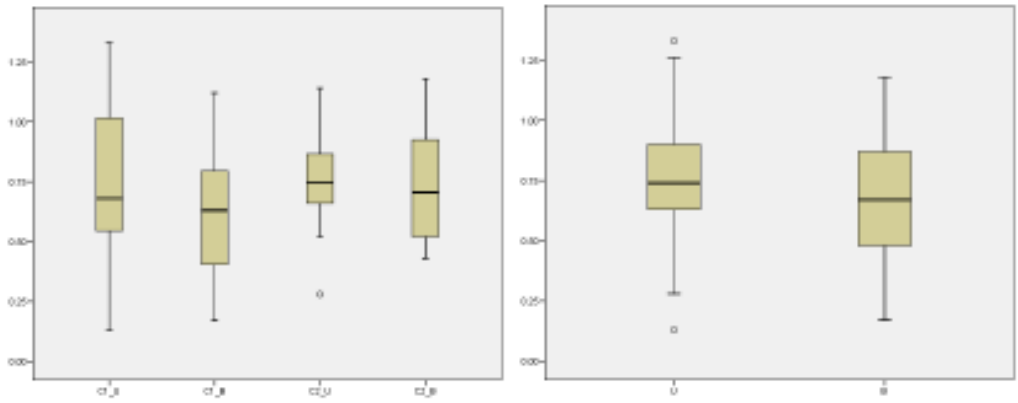
The *Score* and *Time Taken* for each question were measured individually. There were two types of comprehension measurement and analysis; *Overall Comprehension Task* and *Comprehension for Modification Task*. The measurement for *Overall Comprehension Task* was obtained by consolidating the total *Score* and the total *Time Taken* for all five questions. The measurement for the *Comprehension for Modification Task* was obtained by considering the *Score* and the *Time Taken* for the question on model modification only, that is, *Question A5*. The respective analysis was then applied on these measurements. The raw data of those measures can be found in the **Appendix A**.

3.6.1.1. Descriptive Statistics and Analysis

The simplest useful numerical description of a distribution consists of both a measure of centre and a measure of spread (Moore et al., 2006). These measures depict how the data are distributed across the sample. The Figure 3.1 below

illustrates these measures for the *Overall Comprehension Task* distribution. Column *Min* shows the minimum values, column *1st Quarter* shows the first quartile values, column *Mean* shows the average values, column *Median* shows the middle values, column *3rd Quarter* shows the third quartile values, column *Max* shows the maximum values, column *Std Dev* shows the degree of variation, and column *N* gives the number of collected data. Rows *Case 1:UML-B* and *Case 1:B* present the *Rate of Scoring* of the respective models for the *Auction System*. Rows *Case 2:UML-B* and *Case 2:B* present the *Rate of Scoring* of the respective models for the *Library System*. The last two rows present the grouped *Rate of Scoring* based on the models used, regardless of the system. In addition to the table, the respective boxplots are also included to illustrate the distributions graphically for easy viewing and comparison. For the left boxplots, columns *C1_U* and *C1_B* present the *Rate of Scoring* of the UML-B and B models respectively for the *Auction System*. Similarly, columns *C2_U* and *C2_B* present the *Rate of Scoring* of the UML-B and B models respectively for the *Library System*. Columns *U* and *B* in the right boxplots present the grouped *Rate of Scoring* of the UML-B and B models respectively.

	Min	1st Quarter	Mean	Median	3rd Quarter	Max	Std Dev	N
Case 1: UML-B	0.13	0.59	0.74	0.70	1.00	1.33	0.33	21
Case 1: B	0.17	0.41	0.60	0.63	0.78	1.12	0.26	20
Case 2: UML-B	0.28	0.68	0.76	0.75	0.86	1.14	0.19	20
Case 2: B	0.43	0.53	0.73	0.71	0.91	1.18	0.23	21
UML-B	0.13	0.63	0.75	0.74	0.90	1.33	0.27	41
B	0.17	0.48	0.66	0.67	0.87	1.18	0.25	41



Note: **C1_U** => Case 1:UML-B; **C1_B** => Case 1:B; **C2_U** => Case 2:UML-B; **C2_B** => Case 2: B; **U** => UML-B (Case 1 & 2); **B** => B (Case 1 & 2)

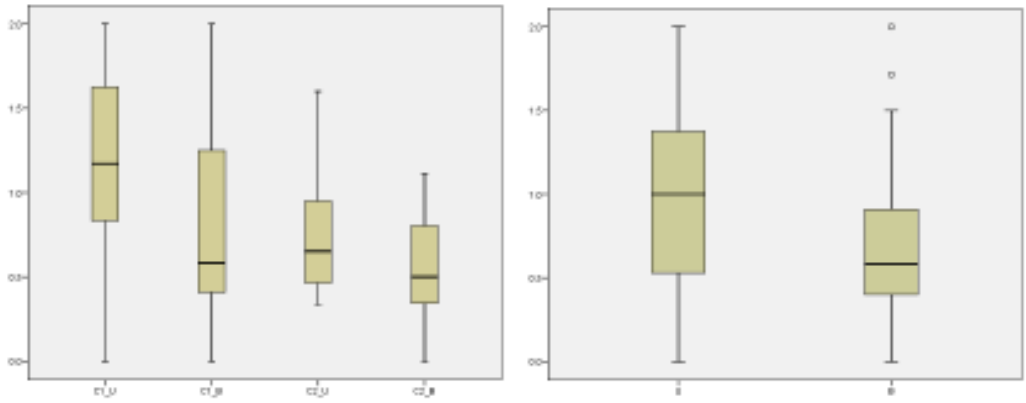
FIGURE 3.1: The Rate of Scoring distribution for Overall Comprehension Task (Unit: marks/min)

Model comprehensibility is the key foundation to efficient maintenance as it assists maintainers to understand system properties correctly and quickly prior to any modification task. Therefore, a modification task was included in this experiment as one of the criteria to assess model comprehensibility. The modification task required the subjects to introduce new features to the existing models. Since the existing models must collaborate well with the new features, the modification task also required some changes to be made on the models. Both of these activities required some understanding of the models.

The Figure 3.2 below illustrates the measures of centre and spread for the *Comprehension for Modification Task* distribution. The description for the columns, rows and boxplots are similar to the Figure 3.1 above. It is important to note that the collected data *N* were twenty-one for *Case 1:UML-B* and *Case 2:B* and twenty for *Case 1:B* and *Case 2:UML-B*, which resulted in forty-one data had been collected altogether for each model. For the modification task however, the data considered for the analysis were slightly less than the collected data. This was due to data cleaning, which was conducted in order to ensure the validity of the analysis. In particular, the analysis excluded the subjects who did not attempt the modification task at all, which numbers are stated in the brackets under the *N* column in the Table 3.4. The excluded data were identified by the zero values (0) in the *Time Taken* for the question on model modification. On the other hand, the subjects who had attempted the modification task for some time (non-zero *Time*

Taken) but failed to get any score (zero *Score*) were included in the analysis. There were two such subjects from the UML-B model and three subjects from the B model, as illustrated in the brackets under the *Min* column. The implication of these data is that the subjects had struggled to understand the model or perhaps had misunderstood the model. Either possibility indicates a problem in comprehending the model. This is the reason why they were included in the analysis.

	Min	1st Quarter	Mean	Median	3rd Quarter	Max	Std Dev	N
Case 1: UML-B	0.00 (2)	1.00	1.20	1.21	1.69	2.00	0.62	18 (3)
Case 1: B	0.00 (2)	0.41	0.80	0.58	1.13	2.00	0.64	16 (4)
Case 2: UML-B	0.33 (0)	0.46	0.72	0.63	0.77	1.60	0.37	19 (1)
Case 2: B	0.00 (1)	0.32	0.59	0.50	0.89	1.20	0.36	21 (0)
UML-B	0.00	0.53	0.98	1.00	1.38	2.00	0.55	37
B	0.00	0.40	0.68	0.58	0.91	2.00	0.49	37



Note: **C1_U** => Case 1:UML-B; **C1_B** => Case 1:B; **C2_U** => Case 2:UML-B; **C2_B** => Case 2: B; **U** => UML-B (Case 1 & 2); **B** => B (Case 1 & 2)

FIGURE 3.2: The Rate of Scoring distribution for Comprehension for Modification Task (Unit: marks/min)

From the descriptive statistics shown in the Figure 3.1 and 3.2 above, it can be seen that the *Rate of Scoring* on the UML-B models is higher than that for the B models. The differences of mean and median values between both models are particularly apparent for the *Comprehension for Modification Task*. These differences may be a reflection of true differences in the population from which the samples were taken. On the other hand, it is possible that the differences may

have occurred by chance in the random samples. In order to assume that the differences obtained from the samples are true differences in the population, the standard statistical inference needs to be applied. The statistical inference is the process of drawing conclusions about the population from the observations about a sample (Gauch, 2000).

3.6.1.2. Statistical Inference and Hypothesis Testing

Traditionally, the most common methods used for inference about means of a single sample, matched pairs or two independent samples are the t procedures or *Student's t-test* (Gossett, 1942). The t procedures are useful in practice because they are robust and quite insensitive to moderate lack of normality especially when the samples are reasonably large (Moore et al., 2006). However, they rely on the use of normal distributions for data where they assume that the dependent variables have normal distributions in the population. Normal distributions have a bell-shaped curve. By definition, the mean and median of such distributions are all equal and ninety-six percent of the data occurs within three standard deviations of the mean. In essence, the t procedures cannot be used if the data are strongly skewed where there are more data on one side of the mean than the other, unless the samples are quite large.

The boxplots in Figure 3.1 and 3.2 above seem to suggest that the distributions are not always normal. A boxplot in general presents a five number summary of a distribution, which includes the smallest observed data, lower quartile, median, upper quartile and the largest observed data. A boxplot can be drawn either horizontally or vertically. The normality of a distribution can be determined in a vertical boxplot such as in the figures by looking at the median, which is demonstrated by the horizontal line inside the box. If the distribution is normal, the median line splits the box into two equal or symmetric parts. Besides, the box is also located in the middle of the vertical line that extends from the top and bottom of the box, which are called *whiskers*. If these patterns are not seen in the boxplot, the distribution is skewed and thus is not normal. Since the samples used in this experiment can be considered as not so large, all these indications may suggest that the traditional t procedures are not so suitable for the data.

At a glance, the data may suggest using the non-parametric methods such as Wilcoxon Rank Tests (Wilcoxon, 1945; Siegel, 1956) and Kruskal-Wallis Test (Kruskal et al., 1952). The advantage of these methods is that they do not require any specific form for the distribution of the population. However, they do not make use of the actual values of the observations. Rather, they work with counts of observations where the observations are classified according to ranks. The tests are applied based on the place in order of each observation in the set of all the data. Because of these characteristics, the methods have been considered as less powerful and less robust than the parametric methods such as the t procedures. Therefore, a method that does not require normality or any specific form of sampling distribution such as in the non-parametric methods but has the same degree of robustness as in the t procedures is highly desired for the data of this experiment.

This experiment employed a robust statistical method called permutation tests for the statistical inference (Efron et al., 1993). The method was chosen as it utilises computing power to relax some of the conditions needed traditionally while at the same time retains the main ideas of statistical inference. The strength of this method is that it does not rely on characteristics of the underlying population distribution and does not require large samples but are capable of generating results that are more accurate than those from the traditional methods (Moore et al., 2006). The idea behind the method is that the original sample represents the population from which the sample was drawn. Many samples are taken from the original sample and the required statistic is calculated from each resample. Thus, the permutation distribution of a statistic generates their respective values from many resamples. The resamples from the original sample represent what would be obtained if many samples were taken from the population. The permutation method uses resampling without replacement technique and therefore, it does not assume random sampling. The method is appropriate if subjects have been allocated randomly to treatments. The permutation method was used in this experiment to calculate the standard errors, confidence intervals and to test the significance level of the observed effects. The analysis was done using the S-PLUS® 7.0 for Windows-Enterprise Developer (Insightful, 2006) software.

The experiment employed a cross-over design. Thus, the analysis of the data needed to be treated differently than the ordinary statistical analysis. The analysis had to consider the period effect and the carry-over effect (Senn, 2002), which are treated independently as follows.

I. Period Effect Consideration (With Outliers)

The analysis for the period effect was performed by firstly obtaining the period difference, which is the difference between the first period's data and the second period's data. While differences between period differences in the same sequence group can be regarded as being random, differences between any two period differences in different sequences would also reflect treatment differences. Therefore, comparing the means of the period differences for the two sequences would allow the treatment effect to be examined (Jones et al., 2003). The first period was when the first treatment or model was given to the subjects. For example, the subjects in *Group X* were given the UML-B model while the subjects in *Group Y* were given the B model. The second period was when the subjects and models were switched. Since the variable of interest of the experiment was the *Rate of Scoring*, the differences in *Rate of Scoring* between those periods were calculated. The two-sample procedure using the permutation method was then performed on the differences by categorising the data according to the model used, namely UML-B versus B.

Prior to applying the procedure however, one must confirm that the generated permutation distribution is close enough to normal. The subsequent analysis should not be performed on the non-normal data, as they would be inaccurate. The Figure 3.3 and 3.4 below illustrate the permutation distributions for the *Overall Comprehension Task* and the *Comprehension for Modification Task* respectively. They confirm that the procedures could be performed, which are based on the normal quantile plots shown on the right side. Any normal distribution produces a straight line on the normal quantile plot, which is a useful tool for assessing normality especially if the histograms appear roughly symmetric and unimodal (Moore et al., 2006). In general, a permutation distribution of a sample mean will always be approximately normal because of the Central Limit Theorem.

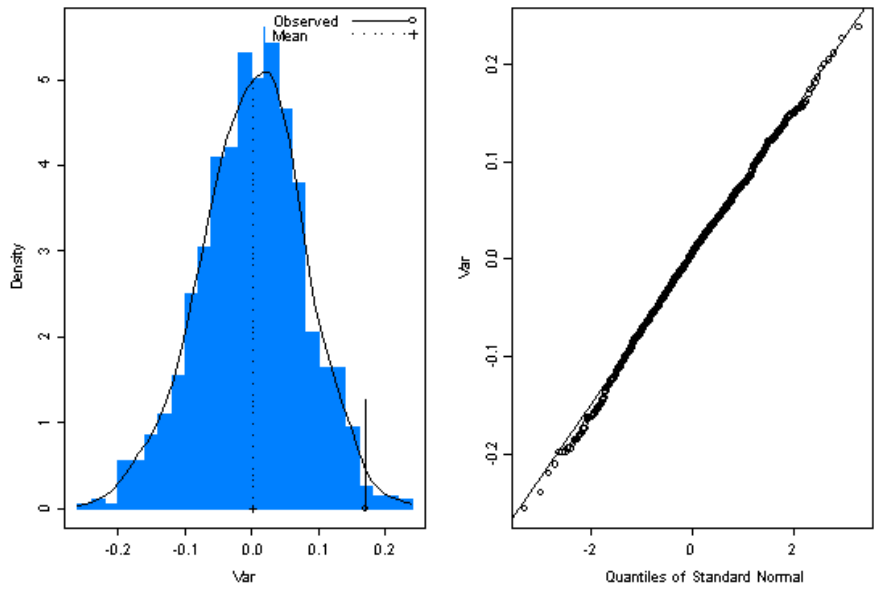


FIGURE 3.3: The permutation distribution for Overall Comprehension Task

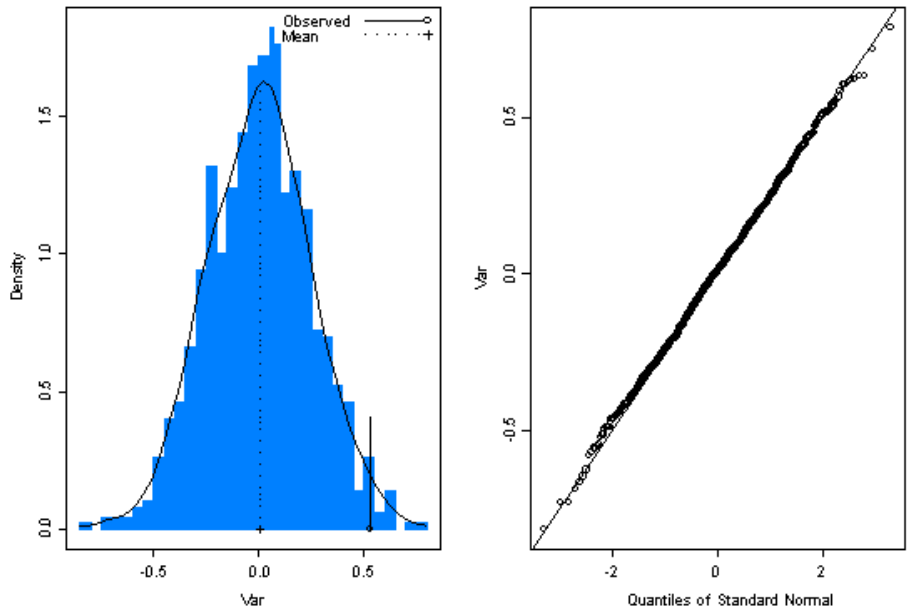


FIGURE 3.4: The permutation distribution for Comprehension for Modification Task

Later, the two-sample procedure was performed. The Figure 3.5 and 3.6 below are the generated permutation test results for the *Overall Comprehension Task* and the *Comprehension for Modification Task* respectively.

```
*** Permutation Test Results ***
Number of Replications: 999

Summary Statistics:
    Observed     Mean      SE alternative p.value
Var   0.1712 0.001153 0.07719   two.sided   0.024
```

FIGURE 3.5: The permutation test results for Overall Comprehension Task

```
*** Permutation Test Results ***
Number of Replications: 999

Summary Statistics:
    Observed     Mean     SE alternative p.value
Var    0.532 0.006809 0.2408   two.sided   0.022
```

FIGURE 3.6: The permutation test results for Comprehension for Modification Task

To understand the implication of these results, attention should be given to the highlighted (bold) values in the figures. The *Observed* entry gives the mean of the sample and the *SE* entry is the standard error, which is the standard deviation for the means calculated by the permutation method. These entries allow the calculation of t-statistics (Mean/Standard error): 2.218 (*Overall Comprehension Task*) and 2.209 (*Comprehension for Modification Task*). The *p.value* entry is the value to be assessed against the significance criterion (α). The *alternative* entry is the indicator on the direction of the test, that is, whether the alternative hypothesis is *greater* or *lower* than the null hypothesis or it is a *two-sided* test with no specific direction.

The generated results above have yet to consider the period effect. They are not quite reflecting the actual results that consider the period effect because the mean period difference for each sequence is an estimate of the difference between two treatments and also between two periods. This means the mean period difference for the "UML-B-then-B" sequence is an estimate of the difference between UML-B and B and the difference between first period and second period. Similarly, the mean period difference for the "B-then-UML-B" sequence is an estimate of the difference between B and UML-B and the difference between first period and second period. In eliminating the period difference or period effect (π) by subtracting the second estimate that is the "B-then-UML-B" sequence, from the

first estimate that is the "UML-B-then-B" sequence, will give an estimate of twice the difference between UML-B and B (Senn, 2002). This is demonstrated by the cross-over model in Table 3.2 where the subtraction of the period differences of the two sequences will give an estimate of twice treatment effect (2τ). In order to overcome this inaccuracy, the values need to be adjusted accordingly. The necessary adjustment is to divide the difference in means and standard errors by two as shown below.

Overall Comprehension Task:

$$\frac{\text{Mean}}{2} = \frac{0.1712}{2} = 0.0856$$

$$\frac{\text{SE}}{2} = \frac{0.07719}{2} = 0.03859$$

Comprehension for Modification Task:

$$\frac{\text{Mean}}{2} = \frac{0.532}{2} = 0.266$$

$$\frac{\text{SE}}{2} = \frac{0.2408}{2} = 0.1204$$

The adjusted means and standard errors above were then used to determine the actual treatment effect that considers the period effect. This was obtained by firstly multiplying the standard errors with the critical values at 95% confidence level from the *Student's t* distribution tables (Geigy 1982; Lindley et al., 1984). The critical values were determined by the degrees of freedom. The degrees of freedom are the number of values that are allowed to vary when calculating the means, which can be calculated manually or using the software. This experiment used the degrees of freedom generated by the software, which were forty for the overall comprehension task and twenty-five for the modification task. After all, both of those methods ended up with quite similar estimates. Later, the values obtained from the multiplication were subtracted from the means to obtain the lowest estimated value for the treatment effect. Similarly, the values were added to the means to obtain the highest estimated values for the treatment effect. The calculated true treatment effect (τ) that considers the period effect at 95% confidence interval for the respective comprehension tasks are shown below. Indeed, they are the estimated differences between the expected *Rate of Scoring* under the UML-B model and that under the B model at 95% confidence interval.

```
Overall Comprehension Task:

0.02 <= τ <= 0.15 (to the nearest 2 decimal places)

Comprehension for Modification Task:

0.06 <= τ <= 0.47 (to the nearest 2 decimal places)
```

To test the significance of the results, the p-values were assessed against the significance criterion (α). Generally, the p-values must be less than $\alpha = 0.05$ for the results to be significant. Unlike the means and the standard errors, the p-values generated by the software previously need not to be adjusted. This is because the p-values are determined from the magnitude of the t-statistics mentioned earlier, which is the ratio of the mean to its estimated standard error. The magnitude of the ratio is the same even for the adjusted values because the division by two is applied to both the means and standard errors. Mathematically, this means no difference in the resulted values. Therefore, the generated p-values can be used straightaway without requiring any adjustment.

Based on the Figure 3.5 above, the p-value for the *Overall Comprehension Task* is 0.024 in a two-sided direction. This means the results of the analysis are significant ($P<0.05$), however with no specific indication on what treatment it favours. Since the research question was to determine whether or not the UML-B model is better than its equivalent B model, a one-sided direction testing was applied. The generated p-value for the one-sided testing is as shown in Figure 3.7.

```
*** Permutation Test Results ***
Number of Replications: 999

Summary Statistics:
    Observed       Mean      SE alternative p.value
Var   0.1712 -0.0006786 0.07826     greater   0.012
```

FIGURE 3.7: The permutation test results for Overall Comprehension Task (one-sided)

The data clearly show that the difference in the treatment effect between the UML-B model and the B model is statistically significant ($P<0.05$). Therefore, the null hypothesis can be rejected in favour of the alternative hypothesis. In other words, it can be concluded that the UML-B model is more comprehensible than

the B model in terms of efficiency in *Overall Comprehension Task*. Subsequently, the same testing was applied on the modification task's data, as shown in the Figure 3.8 below.

```
*** Permutation Test Results ***
Number of Replications: 999

Summary Statistics:
   Observed     Mean     SE alternative p.value
Var    0.532 0.008663 0.2301     greater   0.011
```

FIGURE 3.8: The permutation test results for Comprehension for Modification Task (one-sided)

Similarly, the data show that the difference in the treatment effect between the UML-B model and the B model for the modification task is statistically significant ($P<0.05$). Therefore, it can be concluded that the UML-B model is more comprehensible than the B model in terms of efficiency in *Comprehension for Modification Task*.

It can be seen that the analysis involved two statistical tests on the same data set. One test was for the *Overall Comprehension Task* and the other for the *Comprehension for Modification Task*. It has been claimed that multiple tests on the same dataset can produce a proportionally large number of statistically significant results by chance (Miller et al., 1981; Courtney et al., 1992). Due to that reason, a method called *Bonferroni* (Keppel, 1991; Rosenberger, 1996) was considered in order to ensure the results obtained are still valid. In essence, the *Bonferroni* method requires the significance criterion (α) to be divided by the number of tests. For this experiment, the *Bonferroni* adjustment is 0.025 (α = 0.05/2) for each test. Since the p-values obtained in the analysis are all less than 0.025 therefore, the results are still considered as statistically significant ($P<0.025$).

In addition to the above findings, the conducted statistical inference and hypothesis testing is also believed to essentially provide evidence for the second theory mentioned in the **Objectives** section. The second theory suggests that the integration of graphical and textual representations is more effective in portraying

information. In many cases, the UML-B model and the B model contain similar textual representation or B notation except that the UML-B model uses the graphical representation of UML in concert with the textual representation to illustrate the semantics. Since other possible factors such as subjects' ability, confounding effects and materials' validity had been randomised and treated accordingly in this experiment, the results may also suggest that the integration is better than the textual representation alone. The qualitative data further supports this finding, which will be explained later.

II. Period Effect Consideration (Without Outliers)

The results of analysis presented above were obtained from the data that contained outliers. Outliers are abnormal data that are further away or numerically distant from the others. The outliers may be caused by operational errors such as measurement instrument or process, or they are simply true data that happen to be different. Outliers can be easily identified using a boxplot where they are labeled as small circles or marks outside the box. From the boxplots presented in the previous section, it can be seen that there are at most two outliers in the data.

The outliers in the data had been scrutinised to identify the possible reasons of their occurrence particularly if there were any operational errors. As far as the marking process is concerned, it was correctly performed. One possible error may be due to the recording of the *Time Taken*. The *Time Taken* was given by the subjects by manually stating the *Start time* and *End time*. Perhaps the subjects had incorrectly state the times and thus affected the *Rate of Scoring*. In addition, the errors may be also due to the experiment materials and environment. However, this possibility was less likely to happen as the subjects had the opportunity to raise the issue during the experiment. In fact, the experiment itself was closely monitored. The outliers may also have been caused by personal ability. Otherwise, the outliers can be considered to truly represent the comprehension performance with respect to the respective treatment.

As there were no firm reasons of why the outliers might have occurred, therefore they were included in the analysis. However, the analysis that excluded the

outliers was also performed. This enabled the results of analysis for data with and without outliers to be compared. In particular, it determined whether the conclusion of the findings would be altered if the outliers were not included. The Table 3.4 below illustrates the means, standard errors, t-statistics and one-sided p-values for the data after excluding the outliers.

Item	Mean of Difference x 2	Standard Error x 2	t-statistics	p-value (one-sided)
Overall Comprehension Task	0.1756	0.0786	2.2341	0.013
Comprehension for Modification Task	0.7813	0.2464	3.1708	0.001

Note: significant at α = 0.05 and α = 0.025 (Bonferroni); x 2 = Unadjusted means and standard errors

TABLE 3.4: The mean, standard error, t-statistics and p-value for Overall Comprehension Task and Comprehension for Modification Task (without outliers)

By comparing the values indicated in the Table 3.4 with the Figure 3.7 and 3.8, it can be seen that they are pointing to the same direction. Specifically, the results of analysis for data with and without outliers suggest that the differences between the UML-B and B models are statistically significant for both comprehension tasks. Therefore, the conclusion of findings described previously is still valid with or without outliers.

The measure of efficiency used in the experiment was a ratio of two direct measures, which may have caused the outliers. However, the outliers in the data are very few and have not caused a conflict in terms of the findings' interpretation. The use of the measure is thus considered as appropriate. Therefore, further statistical analyses such as multivariate analysis of variance between the two direct measures and the two treatments are seen as unnecessary.

III. Without Period Effect Consideration

Besides obtaining the treatment effect that considers the period effect, perhaps the analysis that ignores the period effect is still worth doing. The rationale behind this action is to allow a comparison to be made between the unadjusted treatment effect and the adjusted treatment effect. Since the period difference was out of the analysis's concern, the calculation for the unadjusted treatment effect was

performed by obtaining the cross-over difference (Koch, 1972), that is, the difference between the two treatments. The difference was obtained by categorising the data according to models, regardless the treatment sequence. As the data were arranged in this way, the matched pairs test using the permutation method was used.

The objective of the analysis was to determine the estimated treatment effect without the period effect consideration. Therefore, no significance test was pursued. The estimated treatment effect (τ) that ignores the period effect at 95% confidence interval is as follows. It can be seen that there is a slight variation in the estimates especially for the *Comprehension for Modification Task*. The period effect for the *Overall Comprehension Task* is indeed too small which has caused the estimate to appear to be the same as the adjusted estimate, after being rounded to two decimal places.

```
Overall Comprehension Task:

0.02 <= τ <= 0.15 (to the nearest 2 decimal places)

Comprehension for Modification Task:

0.05 <= τ <= 0.52 (to the nearest 2 decimal places)
```

IV. Carry-Over Effect Consideration

The analysis discussed so far concerns the period effect. There is another effect that may influence the results obtained in a cross-over trial, namely the carry-over effect. Although the existence of the effect is admitted, to carry out tests for it is not advisable (Senn, 2002; Jones et al., 2003). This is due to some conflicting statistical theories, which seem to suggest that the tests for carry-over effect are useless. Moreover, it has been shown than if slightly more realistic forms of carry-over test apply, using the models and associated designs can actually be worse than doing nothing at all about the carry-over effect (Senn, 1992).

One possible approach to dealing with carry-over is that of using a wash out period. The wash out period is when the subjects of cross-over trial are given a break to refresh the effect of the first given treatment. When a wash out period is

employed, it is assumed that the effect of the first treatment has disappeared. The measurement taken in the second treatment therefore will not be affected by the first treatment. As far as the experiment is concerned, there was a break between the two sessions. Due to time constraint however, the break was only a few minutes. The implication of this is that the break might have been insufficient for the first treatment's effect to disappear. On the other hand, one can never be certain whether or not the wash out period meets its purpose even if a longer time is allowed. This is particularly true especially when the human's mind and experience sustainability is concerned, which even vary from one person to the other. For instance, it can never be sure that the first treatment effect will disappear if the second session of the experiment is performed after a month or a year. In fact, this approach may introduce other confounding factors that might influence the results due to lack of controls that could be imposed on the subjects. For example, some subjects may be exposed to new knowledge so that the comparison of the two treatments is affected by uncontrolled variables. The subjects' skill in using the methods may also be improved by practice.

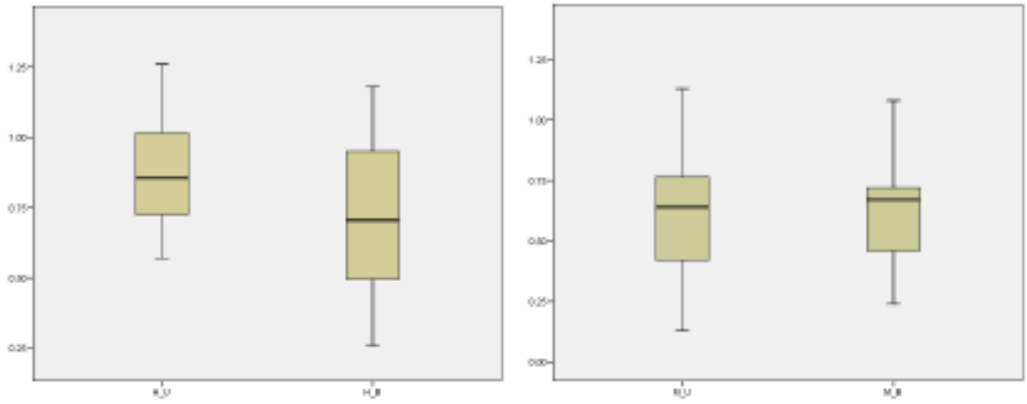
Another possible approach to the carry-over problem is to replicate the experiment several times with different designs. If several experiments with different designs investigated the same treatments and revealed similar results, one could be certain that the results are valid whether or not the carry-over has existed.

V. Unbalanced Randomised Blocking and Cross-over Design

To reduce variability across the two groups, the subjects were allocated randomly based on their knowledge and expertise in object-oriented technology (OO) and formal method (FM). This was based on the subjects' grades on the respective courses during their earlier studies. There were three blocks of ability: *High*, *Moderate* and *Low*. *High* ability refers to subjects who performed very well in both OO and FM courses (Grade B and above), *Moderate* ability includes subjects who performed fairly well either in OO or FM or both (Grade C and above) and *Low* ability comprises subjects who performed poorly in both courses (Grade D and below).

The results of analysis presented so far are based on the assumption that the allocation was a simple random. This is because the cross-over trial as described by Senn (2002) does not require blocking for individual differences. This is shown in the Table 3.2 above where individual differences, represented as the mean effect for each individual (μ), are removed when the results for an individual in each period are subtracted. The treatment distributions for each ability block were then examined. This was to ensure that the direction of treatment effect, UML-B is better than B, is consistent across the three different blocks. This would support the results discussed earlier. The Figure 3.9 below shows the numbers of subjects in each block for both groups. The last two columns present the means and medians of grouped overall *Rate of Scoring* for UML-B and B regardless of the treatment sequence. Boxplots are also included to illustrate the distributions graphically for easy viewing and comparison. Since there were only two subjects for the *Low* ability block, no boxplot is displayed.

Ability	Number of Subjects (Group X)	Number of Subjects (Group Y)	Total	UML-B Grouped Mean/Median	B Grouped Mean/Median
High	9 (43%)	11 (55%)	20	0.87/0.86	0.72/0.71
Moderate	11 (52%)	8 (40%)	19	0.63/0.64	0.63/0.67
Low	1 (5%)	1 (5%)	2	0.63/0.63	0.35/0.35
Total	21	20	41		



Note: **H_U** => High Ability:UML-B; **H_B** => High Ability:B; **M_U** => Moderate Ability:UML-B; **M_B** => Moderate Ability:B

FIGURE 3.9: The Rate of Scoring distribution for different ability blocks (Unit: marks/min)

The figure depicts that UML-B is better than B for the *High* and *Low* ability blocks. However, both treatments appear to be very similar in the *Moderate* ability

block. Since the numbers of subjects for each block are less than thirty, no significance test was pursued. Overall, this observation seems to suggest that the direction of treatment effect is consistent, at least in two out of three blocks.

3.6.2. Qualitative Measures and Analysis

The qualitative measurement was included in the debriefing questionnaire. It was intended to support and explain the quantitative results by providing qualitative insight. Four main aspects were measured. This included the strategies used by the subjects to answer each question, direction and breadth of comprehension employed by the subjects when understanding both models and the preference of model by the subjects. In addition, the subjects were also given the opportunity to provide any personal comments on the models. The details of the questions can be found in the **Appendix A**.

3.6.2.1. Strategies in Answering Questions

This aspect was included to confirm that the answers given by the subjects were derived from reading the model itself. In addition, it was also meant to identify any other strategies that the subjects employed when deriving the answers. For each question, the subjects were asked to indicate whether the answer given was based on *Guess*, *Previous knowledge*, *Common sense*, *Reading the model* or *Other*. The subjects could select more than one selection. If *Other* was selected, they were required to state how the answer was obtained. The Table 3.5 below illustrates the distribution of answers for every question in both cases respectively.

	Guess		**Knowledge**		**Common Sense**		**Reading the model**		**Other**	
	C1	**C2**	**C1**	**C2**	**C1**	**C2**	**C1**	**C2**	**C1**	**C2**
Question A1	6%	28%	46%	29%	30%	17%	18%	13%	0%	13%
Question A2	10%	2%	5%	1%	5%	9%	80%	88%	0%	0%
Question A3	1%	0%	2%	0%	2%	5%	95%	95%	0%	0%
Question A4	0%	0%	11%	0%	11%	10%	78%	90%	0%	0%
Question A5	2%	5%	11%	8%	14%	7%	73%	80%	0%	0%

Note: C1 = Case 1 = Auction System; C2 = Case 2 = Library System

TABLE 3.5: The distribution of answering strategies for each question

It can be seen from the data that majority of the answers were derived from the models since all questions except *Question A1* indicate high percentages under the *Reading the model* column. It is not surprising to discover that *Question A1* of both cases have a variety of answers as the questions were about the meaning of the symbols used in the models, which can be obtained in various ways. For instance, the subjects may already know the symbols from previous experience of learning and using the notations. Thus, the subjects can simply state the meaning without having to refer to the models. In addition, it is worth noting that several subjects used other strategies for *Question A1* in *Case 2* as indicated by *13%* in the table above. As the questions asked about a very specific symbol used in each model namely *$* and *!*, the subjects used the available notes to find the meaning. This can be considered as normal since stakeholders in general should not be assumed to memorise the symbols used especially for newly invented notations. In fact, they constantly refer to handbooks and documentation to accomplish their tasks. This is the reason why the experiment was conducted as an open book exercise, where the subjects were allowed to refer to textbooks or notes.

Another indication that can be seen from the data is that *Question A4* and *Question A5* have some prominent figures under *Knowledge* and *Common Sense* columns, especially for *Case 1*. It is believed that these questions were indeed required the subjects to read the models to obtain the answers. For instance, *Question A5* required model modification, which is impossible to do without firstly reading and understanding the models. The possible explanation for this phenomenon is that the subjects might have combined their understanding about the model with previous knowledge and common sense to derive the answers. The *Auction System* was particularly involved probably because of the problem domain itself. Unlike the *Library System* that is very common, the operations in the *Auction System* are more logical if the subjects could visualise how the operations actually work in the real world. Thus, common sense and previous knowledge dealing with such a system play the roles in this context.

The above phenomenon however is not believed to cause the analysis to be less valid as the majority of the feedback reflects that the models were read during the tasks. Even though the subjects stated other strategies, they still included *Reading*

the model as one of the strategies. Besides applying any other strategy such as *Common Sense* or previous *Knowledge*, the subjects actually combined it with the understanding of the model to derive the answers. In general, the feedback suggests that the questions had been constructed in such a way that they were close enough to the models. In other words, they were relevant to the object of interest, that is, model comprehensibility.

3.6.2.2. Direction and Breadth of Comprehension Strategies

The second aspect of the qualitative measurement was included to investigate the direction and breadth of comprehension employed by the subjects when understanding both models. In particular, the research aimed to find out whether the subjects employed the *Top-down* or *Bottom-up* strategy for the direction, and whether they employed the *Systematic* or *As-needed* strategy or both for the breadth. The direction concerns the *Overall Comprehension Task* while the breadth concerns only the *Comprehension for Modification Task*.

The subjects were not introduced to the terms used for the comprehension strategy, that is, whether it was *Top-down* or *Bottom-up* and *Systematic* or *As-needed* strategy. Rather, the terms were illustrated in specific scenarios that described the actions that the subjects would possibly take in accomplishing the tasks. The descriptions were given in layman terms without introducing any technical jargons. This was to avoid any confusion caused by the jargon and to avoid any unnecessary mental burden imposed on the subjects. The Figure 3.10 and Figure 3.11 below show the proportions of strategies employed in both models respectively.

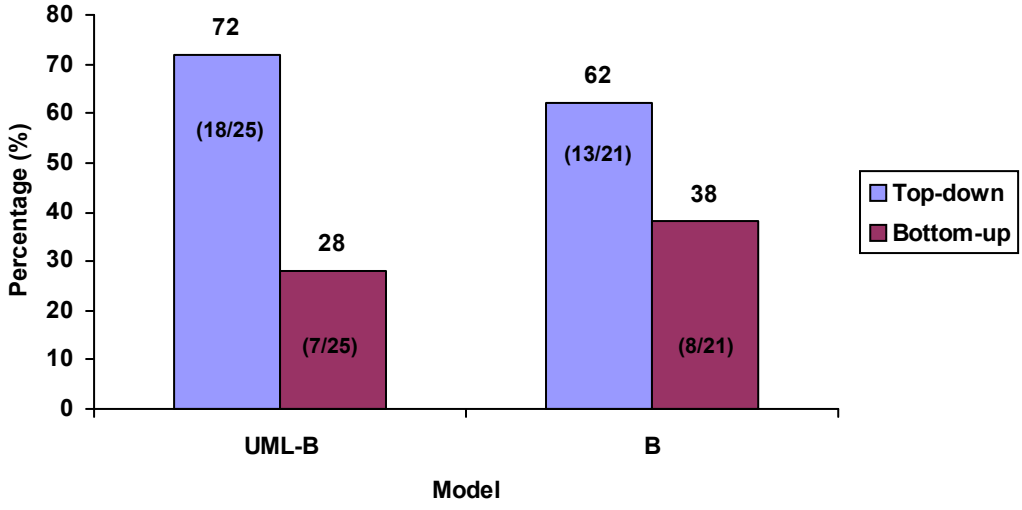


FIGURE 3.10: Direction of comprehension for Overall Comprehension Task (in %)

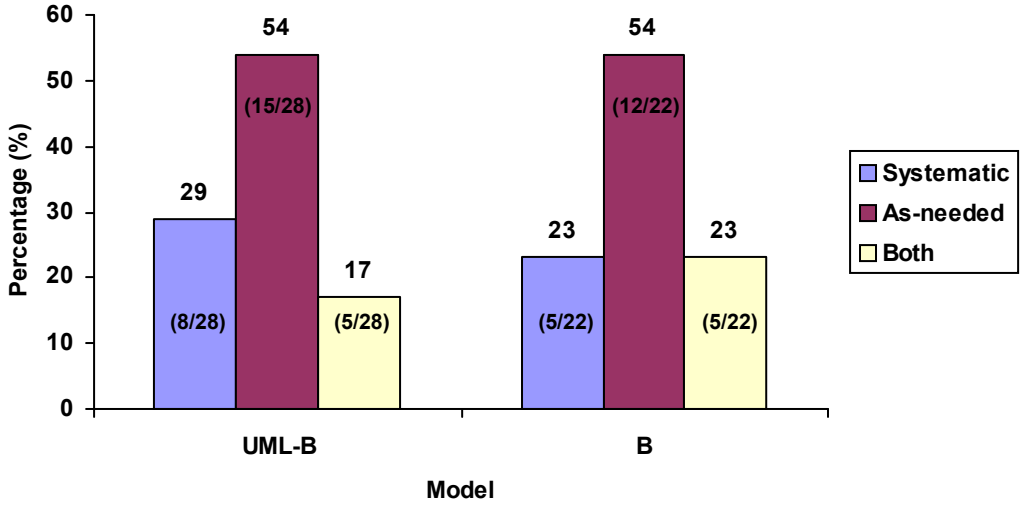


FIGURE 3.11: Breadth of comprehension for Comprehension for Modification Task (in %)

Based on the figures shown above, it seems that the majority of the subjects employed the *Top-down* strategy for the *Overall Comprehension Task* and the *As-needed* strategy for the *Comprehension for Modification Task*. Similar trends can be seen in both UML-B and B models where the differences between models had been found to be insignificant (Chi-Square test: x^2=0.17 for *Overall*, x^2=0.31 for *Modification*; P>0.05).

The *Top-down* strategy may have been employed because the subjects were familiar with both problem domains. These results support the theory that suggests stakeholders use a top-down, goal-oriented or hypothesis-driven approach to

comprehension when they are working in a familiar domain in which they recognise a large number of plans (Shaft et al., 1995; von Mayrhauser et al., 1996). On the other hand, a number of subjects employed the *Bottom-up* strategy, which is about one-third to half of the subjects who employed the *Top-down* strategy. This is possibly because the subjects were quite new to the notations used in the models, as supported by the theory that claims stakeholders with less accumulated notation knowledge and knowledge of the model, spend some time in *Bottom-up* comprehension strategy (von Mayrhauser et al., 1997). This assumption was in fact supported by the subjective comments highlighted by the subjects. For instance, a number of subjects mentioned that the experiment was a good exercise because as it was the first time they were exposed to the application of B notation in real problems with reasonable size.

The Figure 3.11 above shows that the subjects employed the *As-needed* strategy for the *Comprehension for Modification Task* in both UML-B and B models. This may be because the task required the subjects to modify the current models by adding new features. Rather than gaining an overall familiarity with the models, the subjects aimed to make the necessary changes as accurately and quickly as possible. They browsed through the models only to find the respective parts that were thought to be relevant to the task. Another factor that might influence the strategy is the size of the system. It has been claimed that the *As-needed* strategy is more applicable to modification tasks for large systems (Koenemann et al., 1991; von Mayrhauser et al., 1995). Since the models used in this experiment could be considered as quite small, this may suggest that the strategy applies to modification tasks regardless of the system's size. Moreover, the *As-needed* strategy may be applied in any modification tasks regardless of the models used, or even the programming technology used such as demonstrated in a study (Corritore et al., 2001). The possible explanation for this phenomenon is that maintenance costs stakeholders time, effort and money. Thus, they tend to make the necessary modification as accurately and quickly as possible, regardless of the system's type and size.

3.6.2.3. Model Preference

The third aspect of the qualitative measurement was the preference of model. At the end of the experiment, the subjects were asked to provide subjective rating for each model's comprehensibility. The subjective rating was a symmetric five ordinal scales from *–2* for *Very difficult to comprehend* to *2* for *Very easy to comprehend*. In addition, they were also asked to provide their preferences of model to work with in future. The Table 3.6 below illustrates the distribution of the subjective rating and the Figure 3.12 shows the percentage of model notations preference. The figures were based on the thirty-four subjects who responded to the questions.

	-2 Very difficult to comprehend	-1 Difficult	0 Neither difficult nor easy	1 Easy	2 Very easy to comprehend	Total	Median
UML-B	2	10	7	12	3	34	0
	6%	29%	21%	35%	9%	100%	
B	1	3	7	17	6	34	1
	3%	9%	21%	50%	17%	100%	

TABLE 3.6: Subjective rating distribution of model comprehensibility

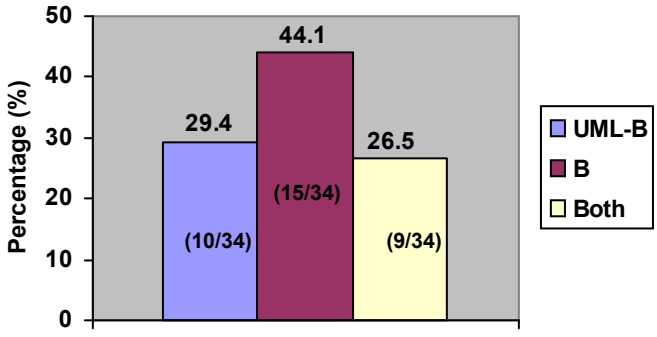


FIGURE 3.12: Preference of model (in %)

The descriptive statistics shown above seem to suggest that the subjects perceived the B model as more easy to comprehend than the UML-B model and the B model was preferred to the UML-B model. In a glance, one may suggest that the B model is better than the UML-B model from the subjects' perspective. However, it is worth noting that the findings may be due to several reasons. Compared to B,

the subjects were only exposed to UML-B a few days before the experiment was conducted. In fact, the lecture session was only about an hour. The subjects therefore had limited experience and time on learning and exploring UML-B. Unlike B, UML-B also lacks comprehensive references with specific examples of application. UML-B is a subset of UML and B, which the subjects have already known. However, there are several integration rules that need to be understood. Thus, it is not surprising to discover that the subjects preferred the B model, as they were more familiar with B. This fact was indeed supported by the informal feedback received from the subjects. Perhaps the perception would be different if more resources were allocated for learning and exploring UML-B, which is worth-investigating in future.

3.6.2.4. Comments on the Models

Besides the above measures, the subjects were also asked to provide personal comments on the models. The UML-B model was perceived as being easy to visualise and understand the scenario more quickly, easy to understand the relationships between operations, easy to develop especially on computers, easy for novices and more logical to developers. The model however was said to be useful only with good tool support. The UML-B model was also commented as being quite 'messy' since the information was scattered around the *Class* and *Statechart* diagrams. In general, the main difficulties of the UML-B model include the interpretation of specific symbols, understanding the integration between the UML diagrams and the B notation, and the tracing between chunks of information. Perhaps these are the reasons why some of the subjects perceived the UML-B model as difficult to understand. On the other hand, the B model was commented as being more formal, less ambiguous and easy to read since the information was kept together as a flow of information. However, the B model was claimed as being harder to develop, lacking visualisation, lengthy and too much text. The hardest parts to understand about the B model are generally about the interpretation of symbols used and the tracing between chunks of information.

3.6.3. Other Findings

3.6.3.1. Absentees and Performance

Another finding seems to suggest that even with very limited training on UML-B, one can still understand the model well. There were eight subjects who did not attend the UML-B lecture and thus depended on the available references or their own knowledge to answer the questions. The *Rate of Scoring* for these eight subjects is shown in the Table 3.7 below. It can be seen that seven out of eight subjects performed better on the UML-B model. Five of these subjects commented that they preferred the B model to the UML-B model. Despite the fact that these subjects disliked and had no training on UML-B, the quantitative measures show that they still performed better on the UML-B model than with the B model. However, the size of this sample is too small to perform reliable statistical significance testing.

Subject	UML-B model	B model	Preference
X08	**0.63**	**0.61**	**U**
X12	**0.63**	**0.53**	**B**
X13	0.64	0.73	B
X16	**0.50**	**0.44**	**B**
X18	**0.66**	**0.48**	**U**
Y01	**0.87**	**0.42**	**U**
Y11	**0.57**	**0.48**	**B**
Y20	**0.77**	**0.71**	**B**

TABLE 3.7: The Rate of Scoring for subjects who were absent during UML-B lecture

3.7. Threats to Validity

Threats to validity are influences that may limit the ability to draw conclusions from the data. In empirical studies, there are four kinds of validity that must be protected from such threats (Cook et al., 1979a; Perry et al., 2000), which are discussed below.

3.7.1. Internal validity

This validity refers to how tight the conditions of the experiment. It concerns whether the experiment is conducted in such a way that there is a clear cause and effect relationship between the dependent and independent variables (Gauch, 2000). In other words, it determines whether the measures of model comprehensibility could be influenced by any other factors than the notations used in the models.

3.7.1.1. Materials

Since the experiment employed a cross-over design, it required two sets of equivalent models to be developed in each case. The equivalency was required not just between two models but also between two cases. Despite the cases must be closely equivalent, they also must be different enough to avoid any significant carry-over effect.

A similar set of questions was prepared for both models to ensure the models were assessed on the same aspects. All questions were exactly the same except one question on the symbols used. This cannot be avoided as each model has its own unique symbols. In addition, the models had similar levels of complexity and size. The difference between models was the notations used. The UML-B model comprised the graphical and textual representations while the B model comprised only the textual representation. In fact, the textual representation in both models was quite the same due to the usage of B notation in both UML-B and B models.

To avoid any comprehension difficulty caused by the problem domains rather than the notations used, two common problem domains were selected namely *Auction System* and *Library System*. Both systems were designed in such a way that they modelled similar operations but in different contexts. For example, both systems involved user accounts administration and business transactions. Although the number of operations in the *Library System* was slightly more than the *Auction System*, it was not seen as a factor that could violate the validity since the operations in the *Library System* were more straightforward. For instance, the borrowing and returning book operations were seen as simpler than the placing bid operations in the *Auction System*. Moreover, some operations in the *Library System* were actually similar except slight changes in the conditions and the effects. For instance, there were two operations on returning a book; one was returning a book on time and the other was late returning. On the other hand, there were also some variations in the questions particularly on the modification tasks in order to reduce any carry-over effect from the first system to the second. The modification task in the *Auction System* involved a task on the user account whereas in the *Library System* involved a task on the business transaction.

3.7.1.2. Maturation

The experiment contained two consecutive sessions where the subjects had to perform two tasks within two hours. This may have caused the subjects to be tired or bored, which affected their performance especially in the second task. To reduce this effect, there was a break between the two sessions and the sessions were made to be as short as possible.

3.7.1.3. Selection of Subjects

The selection of the subjects in the experiment may have influenced the results. Although the subjects were randomly assigned to treatments, the subjects themselves were not quite a random sample. The subjects were students in the university where the research was conducted. They might have been obliged to perform well only because they were taught on the methods. Besides, they might have wanted to impress the lecturers and to make the results looked good. Since

the participation was on volunteer basis, the subjects might have been more motivated and were more enthusiastic about trying new technologies.

Despite those facts, the selection of these subjects was seen as appropriate. This was because as new technologies, it was difficult to get people from the population who had the knowledge of both the UML-B and B methods. Even if there were, those people were academicians or researchers who had personal interest on the methods in many ways. This could make the results to be bias. To reduce the above effects, the subjects were advised to act as naturally as possible and to do their best. In fact, the experiment acted as a revision thus the subjects treated it as learning rather than assessing methods.

3.7.2. External validity

This validity refers to the ability to generalise the results and conclusions of a study to other populations and conditions than those used in the study (Gauch, 2000). In other words, the external validity involves establishing the domain to which a study's findings can be generalised (Yin, 2003). Indeed, internal and external validity are related. Internal validity addresses observing causal relationships under certain conditions, while external validity addresses the domain within which these causal relationships are correct.

3.7.2.1. Students as Subjects

The subjects who participated in the experiment may have not represented software professionals. Ideally, it would be better to have professionals to be involved in the experiment. For pragmatic consideration however, having students as subjects is the only viable option for laboratory-based experiments. It is not always possible to require professionals to spend their precious time trying a new technology unless there is clear and enough evidence that the technology would bring a benefit to their business. At the point when the experiment was executed, there was no empirical evidence on UML-B other than the ones obtained from isolated case studies, which were run by people who had strong interests on its development (Petre et al., 2001; Snook et al., 2003a; 2003b; 2004d; Krupp et al.,

2004; Cansell et al., 2004). In fact, UML-B is a technology that has not yet been used in practice and hence there is no formal training available to professionals. The experiment was the first and only independent assessment with trained potential users conducted on UML-B.

In addition, excessive organisational bureaucracy needs to be undergone to obtain the authorisation to use the professionals. The process is not straightforward unless there is a mutual agreement between academia and industry for work collaboration. This is the main reason why many studies in software engineering use students instead of professionals even though the objective is to draw conclusions for professionals. For software comprehensibility alone, there are a number of studies that use students as the subjects for various objects of interest (Finney, 1996b; Finney et al. 1999; Kamsties et al., 2003; Snook et al., 2004c; Torchiano, 2004; Glezer et al., 2005; Carew et al., 2005). To address the issue however, several studies have attempted evaluating the difference between students and professionals in carrying out judgement task (Höst et al., 2000; Runeson, 2003). The studies have found that that the differences are only minor and concluded that students may be used instead of professionals particularly when the pedagogical goals of the course and the research goals are harmonised. Moreover, students are the next generation of professionals (Kitchenham et al., 2002a). Particularly for new technologies such as UML-B, students are indeed much better trained than most professionals.

There are several benefits that researchers, students and industry could gain from studies with students (Carver, 2003). Among others, the use of students allows the researcher to control factors such as vast variation in expertise and experience among professionals from the object under investigation. The students on the other hand could be exposed to new technologies which are more cutting-edge than those taught in the courses. They could also be introduced to the idea of validating new technologies empirically through experiments. Moreover, the industry is provided with preliminary evidence of the efficacy of new technologies. When some understanding has been obtained from these small-scale and less expensive studies, more extensive and carefully planned studies using professionals could then be invested.

Several assumptions have been made on the population for the experiment's results to be generalised. In particular, it is assumed that the professionals have limited knowledge on at least one of the notations, UML or B or even both. This means even though the professionals are experts on UML, they still need some training on B and vice versa. Based on the knowledge of UML and B, the population could grasp the concepts of UML-B. These conditions which were satisfied by the subjects in the experiment are seen as typical in the real world since professionals are generally specialised in certain expertise and have limited hours for learning any new technology. The expected difference between professionals and students would be perhaps the professionals could perform the comprehension task slightly faster than the students due to experience, provided they use the notations in a similar manner.

3.7.2.2. Toy Problem

The problem domains used in the experiment were not large and may have not been representative of real software systems. As far as the comprehension task is concerned, the systems contained the necessary elements to be assessed. Besides, since the technology is new and the experiment was the first attempt to evaluate it empirically, small and less complex systems were seen as more suitable. Otherwise, the conditions and effects may become too difficult to control and distinguish from one to the other. More control exerted over an experiment is gained only at the expense of its realism (Tiller, 1991).

It is agreed that the effectiveness of maintenance process is best evaluated by considering the entire maintenance process, not only the front-end. The experiment considered only the modification made on the model or specification. This is because to conduct a laboratory-based experiment on such a scale is not practical. Besides, like any other engineering, documentation is important in software engineering (Sommerville, 2001). It is assumed that before any changes are made to the code, the specification must firstly record the changes. After all, the specification is the document that will be referred by stakeholders. If the rate of comprehension gives a positive indication such as demonstrated by this

experiment, one may foresee that the entire maintenance process could also be expedited.

3.7.3. Construct Validity

This validity concerns the establishment of correct operational measures for the concepts being studied (Yin, 2003). The concept being studied in this experiment was the model comprehensibility in terms of notations used. In essence, it is necessary to ensure that the dependent variables are the valid measures for the model comprehensibility.

3.7.3.1. Dependent Variables

There is a concern whether the *Score* or correct answers given by the subjects really portray their true understanding of the models. It is reasonably correct to assume that the subjects will only give correct answers if they understand the objects being asked and the elements illustrated by the models. As the subjects were motivated to perform the tasks, there is no sensible reason for the subjects to give incorrect answers unless they really do not know the answers or have misunderstood the questions or the models. To avoid misunderstanding caused by questions, the questions were made simple and easy to understand. The clarity of the questions had been assessed in the pilot study.

The focus of the experiment was accuracy over time, namely efficiency. Therefore, the *Time Taken* was considered together with the *Score* to determine the *Rate of Scoring*. These measures are valid because a higher *Rate of Scoring* reflects higher efficiency in understanding the model and thus accomplishing the related tasks. On the other hand, the *Rate of Scoring* is an indirect measure which combines two direct measures. The two direct measures may influence each other and contribute to unexpected anomalies. It can also mislead the interpretation where a low score in a short period may be judged better than a high score in a long period. For example, Subject A who scored 20% in 15 minutes (1.33 marks/min) would have a better *Rate of Scoring* than Subject B who scored 95% in 75 minutes (1.27 marks/min). However, the experiment regarded subjects such

as Subject A had chosen not to attempt the questions completely and thus a shorter time was spent. They would have scored higher if they did and spent as much time as Subject B. On the other hand, it is logical for Subject B to score higher due to the amount of the time spent. As far as the efficiency in understanding is concerned, Subject A is still considered as better than Subject B even with a lower score because he or she could obtain 1.33 marks in one minute. Furthermore, the experiment had a specific time frame. Subjects seemed to consume the given time and thus spent about the same time whether or not they had answered all questions. Thus, such possibilities have been found to be low. Moreover, based on the distributions illustrated in the Figure 3.1 and 3.2 above, very few anomalies are observed (no more than two outliers). Therefore, the use of *Rate of Scoring* as the measure of interest in the experiment is acceptable.

The experiment was conducted in a specific duration of time. Rather than letting the subjects to use as much time as they required, they were given a specific time frame to attempt the questions. This may have imposed time constraints to the subjects, which affected their answers. On the other hand, the subjects would be likely to engage in mental elaboration process if there was no time limit. They would possibly keep restructuring the built knowledge structures or mental models. This means their answers would be much influenced by prior knowledge rather than what were being presented.

3.7.3.2. Qualitative Measures

Some of the qualitative measures were retrospective. Therefore, there was a risk that the subjects responded based on what they thought they did rather than what they actually did. Giving the debriefing questions as closely as possible to the tasks reduced this threat, as the subjects were still aware of what he or she was doing.

3.7.3.3. Marking Process

Even though the questions were mainly open-ended, the answers were expected to be exact due to the formality imposed by the models. The marking process was

quite straightforward by looking for certain keywords in the answers. There was no personal information such as student's name on the answer sheets. Therefore, the answer sheets were anonymous to the researcher during the marking. Because different models use different notations, the researcher cannot avoid being aware of the model being marked. Although this may pose a threat to the validity of the experiment however, it can be regarded as negligible as the researcher is an independent user of both methods.

3.7.3.4. Time Recording

The duration of comprehension task was determined based on the starting and ending time given by the subjects. As the process was manual, there was a risk where the subjects had inaccurately specified the times. Moreover, there may be cases where the subjects had been dishonest where they stated incorrect times on purpose. To reduce this effect, the subjects had been reminded to state the times accurately. They were asked to treat the experiment as a revision and were assured that no personal assessment would be made. They therefore were advised to be honest and do their best.

3.7.4. Conclusion Validity

This validity concerns the ability to draw the correct conclusion about relations between the treatment and the outcome of an experiment (Wohlin et al., 2000)

3.7.4.1. Heterogeneity of Subjects

The subjects might have different ability, experience and degree of training on the notations, which may influence their understanding of the models. There was a risk that the variation due to individual differences might have affected the results. On the other hand, more homogeneous subjects would affect the external validity since they were not selected from a general enough population. This factor was reduced by firstly selecting the subjects from the same course. The subjects were third-year Undergraduate and Masters students of Computer Science and Software

Engineering. In general, they were taught on UML and formal methods at some points of their studies. However, the degree of usage and experience in applying the notations vary. Therefore, the blocking and random allocation techniques were applied to reduce the subjects' variability across groups. These techniques should distribute such variability between the two groups. As in any sampling method, there is always the chance that an unexpected allocation has occurred, which cannot be eliminated entirely.

3.7.4.2. Familiarity of Subjects

The subjects were taught formally on B for about nine hours and one hour on UML-B. The training on B was much longer because some principles used in UML-B are also based on B. The one-hour session of UML-B was intended to introduce how to integrate UML and B notation. The results may have been different if the amount of training was much longer. However, it was believed that the allocated training time was appropriate to test the effect and was quite realistic for practitioners to adopt.

3.8. Conclusions and Future Work

This chapter has presented an experiment conducted on a UML-B model and a B model. In particular, the experiment assessed the notations used in the models. The objective was to explore whether the notation used in the UML-B model is more comprehensible than the notation used in the B model. The model comprehensibility was measured based on the subjects' efficiency in understanding the notations used in the models and performing the required tasks.

The findings indicate that the integration of both semi-formal and formal notation is useful in promoting model comprehensibility as compared to the formal notation alone. A model that integrates the use of both notations such as UML-B is capable of expediting the subjects' comprehension task with accuracy even with limited training. The model allows the subjects to grasp the required information

more quickly and use it to perform the subsequent tasks correctly. This finding is appealing as it suggests that introducing some graphical features of a semi-formal notation into a formal notation significantly improves the formal notation's accessibility. Besides allowing the formal notation to be more understandable, the graphical representation seems to make its daunting mathematical features interesting and approachable.

The findings also seem to support the theory that suggests the integration of graphical and textual representations is more effective in portraying information. In many cases, the UML-B model and the B model contain similar textual representation in the form of B notation except that the UML-B model uses the graphical representation of UML in concert with B to illustrate the semantics. Since the possible confounding factors had been randomised and treated accordingly in this experiment, the results suggest that the integration is better than the textual representation alone. As far as the experiment is concerned, this theory helps to explain why the UML-B model is more comprehensible than the B model.

The findings of the experiment indicate that one can still comprehend the notation in a UML-B model even with very limited hours of training. However, the underlying assumptions about the population that is represented by the sample should be understood. Practitioners should only be expected to perform well on the UML-B model if they have been exposed to both UML and B. In addition, basic understanding of the theoretical aspects of formal method and object-oriented technology is also seen as necessary for promoting the comprehension.

For another aspect of evaluation, the experiment also provided some qualitative evidence on the strategies used during the comprehension task. It has been found that both notations in the UML-B model and the B model require the users to apply the *Top-down* strategy rather than the *Bottom-up*. The qualitative data also show that the users use the *As-needed* strategy when modifying both models. These findings imply that the difference in the notations does not influence the way the models are understood. Nevertheless, this knowledge could act as a basis in improving the methods in future. For instance, knowing that people tend to use

the *Top-down* strategy in understanding a UML-B model, the inventor may utilise the upper-level parts more to illustrate the important elements. The upper-level parts of a UML-B model comprise the graphical features of UML diagrams. Thus, perhaps more functionality could be introduced to the diagrams. One possible advantage of this is that the complexity caused by the B notation could be reduced. This also means that the tools which accompany UML-B should support the transformation of more graphical representation to equivalent B textual representation.

There are several ways in which the experiment and its findings could be improved further. It has been pointed out that the hallmark of good experimentation is the accumulation of data and insights over time (Basili et al., 1986; Tichy, 1998; Basili et al., 1999; Jeffery et al., 2002). Therefore, one possible way of improvement is through replication, where the experiment will be repeated on different samples of the population with slightly different conditions and design. This would help in determining how much confidence can be placed in the results of the experiment (Basili, 1992). As the objective of a specification is to further stakeholders' understanding of a problem domain, the investigation of the notation will be extended to include the resulting cognitive model developed by the stakeholders. The measurement will not only assess the notation's ability to present information that can be understood but also its ability to facilitate the construction of problem domain knowledge. The efficacy of UML-B will be further investigated by assessing its model development process through surveys. The qualitative methodology such as found in the social sciences will be employed to gather a holistic understanding of the important factors and how and why they may influence the effectiveness of the method. The qualitative approaches also allow the users' perception towards the method's ease of use to be explored and better understood.

In general, the experiment has met its main objectives. Despite several validity issues such as discussed in the **Threats to Validity** section, it is believed that the findings of the experiment are useful to the software engineering community in many ways. Instead of relying on the inventor's assertions, the practitioners are now provided with some preliminary empirical evidence, which is based on an

independent study. In essence, conducting a perfect experiment in software engineering field is virtually impossible as it involves great coordination between product, process and people in a strictly controlled situation. However, as long as the assumptions and limitations of the study are communicated clearly, the findings could be implied and channelled by the community to the right way.

Chapter 4

Measuring the Usability of the UML-B Method

Modelling is vital in the development and maintenance of software systems. It allows the characteristics of the existing and future systems to be captured and understood. The modelling process produces models where the requirement specification is one of them. Software requirement specification is a conceptual model that establishes the connection between the user's needs of a system and the software solution to meet them. It is an abstract, clear, precise and unambiguous conception of a system, which is developed by using the appropriate notations. In essence, it provides the material support for recording and communicating all the relevant aspects of a problem domain (Motschnig-Pitrik, 1993; Loucopoulos et al., 1995).

A conceptual model is produced through the use of a designated modelling language or notation. Some examples of the existing notations include semi-formal notations such as Entity-Relationship Diagram (ERD) (Chen, 1976; Elmasri et al., 2001), Data-Flow Diagram (DFD)(Yourdon, 1989), Unified Modelling Language (UML) (OMG, 2006) and Open Modelling Language (OML) (Firesmith et al., 1998), and formal notations such as Z (Spivey, 1992, Bowen, 1996) and B (Abrial, 1996). In addition, there are also notations that integrate both semi-formal and formal such as UML and Z (Dascalu et al., 2002; Martin, 2003; Kim et al., 2004), and UML and B (Shore et al., 1996; Sekerinski et al., 1998; Meyer et al., 1999; Ledang et al., 2002; Lano et al, 2004, Snook et al., 2006).

A modelling notation provides a set of constructs from which the conceptual model is derived. Each construct has its own well-defined syntax and semantics that determine which aspects of a system it can represent. For instance, a *Class* diagram in UML is used to illustrate the static structure of a system where it shows the entities involved and the relationships between them. On the other hand, a *Statechart* diagram is used to portray the behaviours of the entities. The specific role played by each construct allows the notation to focus on certain perspectives that are considered as important to the system being built. However, it also imposes some constraints that not only limit the expressiveness of the model but also affect the usability of the modelling notation.

The usability of modelling notation is one of the factors that determine the quality of a model. It is impossible to achieve a high quality model or even produce a model if the notation in the first place cannot be learned, used and understood by users. It has been suggested that the understandability, learnability, operability and attractiveness of a software entity such as notation used should be assessed to determine its usability (ISO 9126-1, 2001). In addition, the ability of a notation to capture the problem domain, the extent to which the notation is formalised to enable execution and how relevant knowledge of the domain may be articulated in the notation (Krogstie, 1998) are also worth-considering when assessing its usability.

In the previous chapter, the comprehensibility of the notation used in UML-B has been assessed based on the results obtained from a controlled experiment. The controlled experiment evaluated the notation comprehensibility in terms of how easy it is to understand a UML-B model from the perspective of users who interpret the model. The results of the experiment suggest that the UML-B model is more comprehensible than the B model. The findings however cannot suggest by any means that the notation is also usable from the perspective of developers who use UML-B for modelling. Neither could they determine whether or not the notation is easy to learn and operate, and suits the developers' common needs and expectations. It is believed that in order to fully understand the strengths and weaknesses of UML-B, another empirical assessment should be conducted to evaluate other aspects of usability especially during model development process.

This chapter presents a survey conducted on UML-B, particularly the notation used. The notation includes the use of *Class* and *Statechart* diagrams of UML and the use of B syntax for expressing constraints and actions on the diagram elements. The survey aimed to assess the usability of the notation from developers' perspective. Usability in this context means the understandability/comprehensibility, learnability, operability and attractiveness of the notation in supporting the modelling process. The assessment was conducted by using a usability evaluation framework namely the Cognitive Dimensions of Notations (CD) (Green, 1989; Green et al., 1996) with the usability criteria suggested in the International Organization for Standardization (ISO) (ISO 9126-1, 2001; ISO 9126-3, 2003; ISO 9126-4, 2004). As usability depends on the notation and its environment, the evaluation included the tools that accompany UML-B namely Rational Rose (Rational, 2000) and U2B (Snook et al., 2004b), whenever appropriate. Rational Rose provides the environment for the UML-B model development while U2B is a tool that generates a B model from a UML-B model so that it can be verified by B tools such as Atelier-B (ClearSy, 2003), B-Toolkit (B-Core, 1999) and Click'n'Prove (Abrail et al., 2003).

The following section explains the technical aspects of the survey's preparation and execution. Section 4.2 and 4.3 present the results and data analysis respectively. Section 4.4 discusses the outcomes and contribution of the survey. Section 4.5 explains several threats to the validity of the results. Finally, Section 4.6 concludes the chapter with a summary of the main findings and future work.

4.1. Objectives and Methodology

4.1.1. Motivation and Approach

The survey was qualitative in nature. Despite the fact that some of the data were quantified using an ordinal scale, the bulk of the analysis was interpretative. This type of analysis was carried out due to the problem at hand, that is, the survey attempted to understand the nature of experience of using UML-B. As little is known about the method, the survey aimed to explore and gain novel understandings of its use through qualitative data and analysis. The analysis allows the intricate details about the phenomena such as feelings, emotions and thoughts to be extracted and learned.

There are many different approaches to dealing with qualitative data employed in the social sciences (Gilgun et al., 1992; Cassell et al., 1994; Denzin et al., 1994; Gubrium et al., 1994; Westbrook, 1994; Morse et al., 1995). The survey adopted one approach, namely the grounded theory (Glaser et al., 1967; 1999; Strauss et al., 1998). There are two variations in the approach, which are based on different directions taken by its originators; Glaser (Glaser, 1992) and Strauss (Strauss et al., 1998). The essential differences between these two variations can be found in the literature (Babchuck, 1997). This survey employed Strauss's approach because it is more systematic and directive. In particular, it contains more formal models and procedures to generate theories. It also encourages a qualitative study to have a research question so that the researcher can stay focused in the midst of masses of data. As a qualitative study, the research question should be broad and open-ended.

The theory in the approach means theory that is derived from data, systematically gathered and analysed through the process. The approach was chosen because unlike the controlled experiment conducted previously, this survey was not based on any specific theory. The approach allows the study to be initiated without a preconceived theory in mind, where the researcher could start with a phenomenon and allow the theory to emerge from the collected data. As the theory is drawn

from data, it is likely to offer insight, enhance understanding and provide a meaningful guide to action (Strauss et al., 1998). It is believed that the theory is more likely to resemble the reality than the theory derived by merging concepts based on how one thinks things ought to work.

The survey followed the scientific approach to studying software engineering phenomena (Jeffery et al., 2002), as illustrated in the Figure 4.1 below. In the approach, the process begins when a researcher becomes aware of a phenomenon that is poorly understood. The phenomenon is investigated using empirical methods and analysis so that a better understanding about the phenomenon can be obtained. The understanding will lead to the formulation of a tentative theory, which is open to testing and evaluation by the software engineering community. The theory that has been evaluated will be independently replicated to prove or disprove the results and if necessary, the theory revision will also take place to improve the theory. The theory can then be applied to improve the phenomenon.

In general, the survey aimed to formulate tentative theories of the usability of integrated methods (semi-formal and formal notations) such as UML-B, based on the understanding obtained from the qualitative analysis using the grounded theory approach. As one single study can never embrace all possible situations, the survey sought to provide some preliminary evidence of such methods' likely strengths and weaknesses when used under certain known conditions. It was also intended to identify any threats that could hinder such methods' usability and any opportunities that could improve them further. The tentative theories can act as a basis for further investigation and analysis in future.

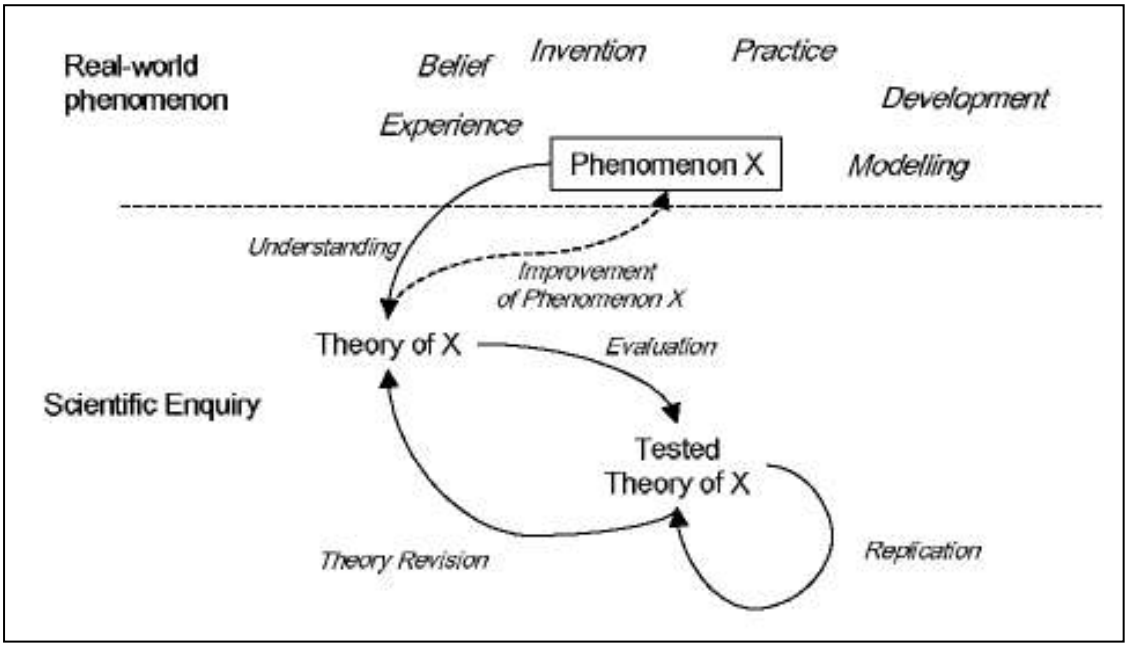


FIGURE 4.1: A scientific approach to studying software engineering phenomena (Jeffery et al., 2002)

One of the subjective comments obtained from the controlled experiment was that the UML-B was seen as easy to develop particularly on computers. The method was also commented to be useful only with good tool support. The hypotheses were given by subjects who dealt with the already developed UML-B model, not modelling. This could not suggest that the hypotheses are true from developers' perspective for modelling purposes. The survey therefore included these hypotheses in its investigation of the phenomenon through the following broad research questions:

Do individuals who develop a UML-B model perceive the method and model as usable (easy to understand, easy to learn, easy to operate, and attractive)?
What are the characteristics of UML-B method and model that affect their usability?

4.1.2. Materials

The survey instrument was developed based on the ideas proposed in the Cognitive Dimensions of Notations (CD) usability framework (Green, 1989; Green et al., 1996). The framework was adopted because it is a tool that aids the

usability evaluation of information-based artefacts (Green et al., 1998). As a usability tool, it captures a significant amount of psychology and Human Computer Interaction (HCI) aspects that focus particularly on the notational design. Moreover, it concentrates on the processes and activities by considering the perspective of people who deal with the artefact and its environment. The framework comprises fourteen dimensions, which acted as the response variables in the survey. The detailed explanation of the dimensions and the framework has been described in **Chapter 2**.

The questions for the survey were constructed by following the proposed CD questionnaire (Blackwell et al., 2000). The advantage of using the standard instrumentation such as proposed by the CD questionnaire is that it has been assessed for validity and reliability by the authors. As the CD framework is widely used by other researchers who are investigating the usability of notations such as UML diagrams (Kutar et al., 2002) and Z (Triffitt et al., 2002), it also provides a mechanism to compare the results of this survey with the results of other similar studies.

The CD questionnaire is intended to present the dimensions in general terms, applicable to all information artefacts rather than presenting descriptions specialised to a specific system under consideration. The questionnaire was therefore tailored and modified slightly to reflect the characteristics of UML-B. The proposed CD questionnaire also employs an open-ended question approach, which could complicate the data analysis. The questions for the survey were thus designed to include a set of answers using an ordinal scale together with the open-ended questions. Besides reducing the data analysis's complexity, it allows the survey to obtain some quantitative measures rather than qualitative measures exclusively.

In addition to the CD framework, the questions of the survey were also constructed based on the usability criteria proposed by the International Organization for Standardization (ISO)(ISO 9126-1, 2001; ISO 9126-3, 2003; ISO 9126-4, 2004); understandability, learnability, operability and attractiveness. There were twenty questions in the survey; fourteen questions reflected the

fourteen dimensions of the CD framework, five questions represented the ISO's usability criteria and one question gathered suggestions for improvement. The questions in the survey were presented in random order without following a specific sequence of dimensions. To ensure the questions were purposeful and concrete, the general guidelines on survey question construction were followed (Kitchenham et al., 2002b, Gauch, 2000).

The questions used an ordinal scale that provided the respondents with five possible levels of agreement such as *–2* for *Very Difficult* to *2* for *Very Easy*. There were also questions that required either *Yes*, *No* or *Not Sure*. The five levels were chosen because they cover the possible categories of the variables. An odd number of levels were used because odd numbers contribute to the achievement of better results as they are balanced (Bonissone, 1982; Godo et al., 1989). Besides the selection on the scale, justification of the answer given was also required such as *Why?* or *Which part?*. This acted as the qualitative data, which were used together with the quantitative data on the scale for the analysis. To give an overview of the questions, below are some examples of the survey questions. The first question concerns the *Visibility and Juxtaposability* dimension. The second question involves the *Hard Mental Operations* dimension. The details of the questions can be found in **Appendix B**.

If you need to compare different parts of your UML-B model (e.g. between diagrams or windows of different operations etc.), how easy is it to view them at the same time in Rational Rose?

Very Difficult Very Easy

-2 -1 0 1 2

Why?

Do you find any complex or difficult tasks to work out in your head when modelling your UML-B model? No Not Sure Yes If Yes, what are they? If No or Not Sure, why?

The CD framework describes the necessary conditions for usability based on the structural properties of a notation, the properties and resources of an environment, and the type of user activity; *Incrementation*, *Transcription*, *Modification*, *Exploratory Design*, *Searching* and *Exploratory Understanding* (Blackwell et al., 2003). In particular, it addresses the question whether the users' intended activities are adequately supported by the structure of the notation used and its environment. For the survey, the identified users' intended activity was *Exploratory Design*, which the users employed UML-B (notation and environment) to design a conceptual model. The survey questions and analysis therefore were tailored to focus on this aspect.

The survey questions were reviewed by a focus group prior to distribution. There were four people involved in the process. The purpose of the review was to identify any missing and unnecessary questions as well as ambiguous questions and instructions.

4.1.3. Participation

Thirteen out of fourteen participants responded to the survey. Three of the submitted questionnaires could not be retrieved and thus were excluded from the analysis. As a result, the questionnaires from ten participants were analysed. Eight participants were from Asia and Europe and the remaining two participants were from the United Kingdom. There were four women and the rest were men.

The participants were Masters students of Software Engineering course at the University of Southampton, who registered for the "Critical System" course in

Spring 2006 (ECS, 2007). They were chosen using *theoretical sampling* due to their potential contribution towards the theory development for integrated methods such as UML-B. Specifically, they were selected because they were taught formally on B (nine hours) and UML-B (one hour) during the course. Basic knowledge of both methods is necessary to develop a UML-B model. Moreover, the participants had some practical experience of using UML-B and its tools when participating in the survey. In particular, they used them to develop a model of a system in one of the coursework at the end of the course. The participants were also involved in the controlled experiment described in **Chapter 3**. The survey was administered a month after the experiment.

The survey adhered to the University's ethical policies and guidance for conducting research involving human participants (UoS, 2007). The participants were aware that the survey was intended for research purposes. They were motivated to participate as it helped them in exploring the method besides providing a space for reflection on the learning prior to the examination.

The subjects were in the final semester of their Masters course. They therefore had reasonable amount of experience and knowledge of software development. Some of them had some work experience. They were the next generation of professionals, thus they represented closely the population under study; software developers.

4.2. Results

The following paragraphs present the responses for each of the questions in the survey questionnaire. The first fourteen questions reflect the dimensions of the CD framework while the subsequent five questions represent some of the usability aspects suggested by the ISO. The last question is comments for further improvement.

4.2.1. Visibility and Juxtaposability

The question **(1)** of the survey assessed the ability of UML-B to allow the user to view every component of its model simultaneously or view two related components side by side at a time. As UML-B resides in Rational Rose, the assessment particularly concerned the ability of the application to support the above user's activities.

The Table 4.1 below shows the distribution of answers for question **(1)**. It can be seen that three of the respondents considered the activities as "Easy" and "Very easy". They commented that navigation in Rational Rose was generally easy as they could view different parts of the model at the same time by opening several windows. For instance, the application allowed them to compare different operations either from one class or different classes simultaneously. They also found that switching around the windows was pretty straightforward.

There were four respondents who regarded the activities as "Neither difficult nor easy", which contributed to the median value. These respondents had a mixture of agreement on the ability of Rational Rose to support the activities. They agreed that the application supported the viewing of different operations, however it did not support the viewing of different models or even diagrams. For instance, they had difficulties in viewing a *Class* diagram and its *Statechart* diagrams at the same time, which made the process of mapping the operations in the *Class* diagram and the transitions in the *Statechart* diagrams tedious. The only solution was to open another Rational Rose application in *Read-only* mode, which was awkward to do.

The remaining respondents considered the activities as "Difficult". Besides the above limitations, they discovered other user-friendliness issues. They found that some common modelling functionality was not visible on the toolbar. For instance, there was no *Aggregation* icon on the toolbar and they had to get it through several intricate steps, which was not obvious. In fact, they found that *Help* feature was not so helpful.

	-2 Very difficult	**-1 Difficult**	**0 Neither difficult nor easy**	**1 Easy**	**2 Very easy**	**Total**
UML-B	0	3	4	1	2	10
Model	0%	30%	40%	10%	20%	100%
& Rose	30%		40%	30%		100%
Median	0					

TABLE 4.1: Distribution of answers for the "Visibility and Juxtaposability" dimension

4.2.2. Viscosity

The question **(2)** of the survey assessed the degree of effort required by the user to perform a change in the UML-B model. The change in this regard includes editing the diagrams and the respective semantics of the model in Rational Rose as well as retranslating the model to a B model by using U2B. The question required the respondents to indicate the difficulty level and state any particular changes that they found difficult or tedious to make.

The Table 4.2 below shows the distribution of answers for question **(2)**. It can be seen that six of the respondents considered the task as "Easy" and "Very easy". This resulted in the typical comment or median as "Easy". Most of these respondents did not state any specific changes that they thought would be difficult. However, two respondents commented that Rational Rose did not support some changes automatically. For instance, if a variable name was changed in the *Class* diagram, the change was not reflected in other parts such as in the *Statechart* diagram or in the semantics where the variable name was used. A similar situation occurred for the variable deletion. Thus, the changes had to be done manually by visiting the respective parts of the model.

The remaining respondents who considered the changes as "Difficult" highlighted other issues such as Rational Rose did not support *undo* and *drag-and-drop* operations. One respondent highlighted that when a deletion was made in the diagram panel, the item would still exist in the model although it did not appear on the diagram. The right way to do the deletion is in the navigation panel, which was not obvious to the respondent.

None of the respondents mentioned any difficulty with U2B.

	-2 Very difficult	**-1 Difficult**	**0 Neither difficult nor easy**	**1 Easy**	**2 Very easy**	**Total**
UML-B Model & Rose & U2B	0	4	0	5	1	10
	0%	40%	0%	50%	10%	100%
	40%		0%	60%		100%
Median	1					

TABLE 4.2: Distribution of answers for the "Viscosity" dimension

4.2.3. Diffuseness

The question **(3)** of the survey assessed the complexity or verbosity of the notation used in UML-B to express a meaning. The notation includes the use of *Class* and *Statechart* diagrams of UML and the use of B syntax. The question required the respondents to indicate how simple to describe what they intended in the model.

The Table 4.3 below shows the distribution of answers for question **(3)**. It can be seen that six of the respondents considered the task as "Simple" and "Very simple". This causes the median to be "Simple". These respondents generally agreed that UML diagrams made the modelling process easier. They started the process by identifying the main objects or entities involved in the problem domain and connecting the entities using the appropriate relationships. The diagrams acted as a base for them to add specification details using the B syntax. These respondents nevertheless admitted that they needed to think in object-oriented way during the process.

Three respondents commented the task as "Neither complicated nor simple". One respondent believed that the diffuseness would depend on the problem at hand. Two respondents thought that the task was not simple due to lack of documentation. The remaining one respondent who thought the task as "Complicated" had difficulty in dealing with UML diagrams and B syntax at the same time. The respondent was confused whether to specify the semantics of the

operations in the specification window of the *Class* diagram or the *Statechart* diagram.

	-2 Very complicated	**-1 Complicated**	**0 Neither complicated nor simple**	**1 Simple**	**2 Very simple**	**Total**
UML-B Model	0	1	3	4	2	10
	0%	10%	30%	40%	20%	100%
	10%		30%	60%		100%
Median	1					

TABLE 4.3: Distribution of answers for the "Diffuseness" dimension

4.2.4. Error Proneness

The question **(4)** of the survey assessed the tendency of the notation to induce mistakes. Since UML-B's notation comprises UML diagrams and B syntax, the questions were divided into two parts. One was meant to assess the diagrams and the other was for the B syntax. The question required the respondents to indicate how easy to make mistakes when modelling the diagrams and defining the formal semantics using the B syntax.

The Table 4.4 below shows the distribution of answers for question **(4)**. It can be seen that six of the respondents considered making mistakes in the diagrams as "Neither difficult nor easy", which contributes to the median value. These respondents agreed that modelling using the diagrams was simple. However, since the diagrams would be translated to a B model at the end, they had to be more careful and conscious. Each time they added a feature to the diagrams, they tended to transform the UML-B model to the B model using U2B to see the effects. They wanted to ensure the added feature had the effect that they intended in the B model, besides being able to verify the model using B tools.

Two respondents commented that making mistakes in diagrams was "Easy" and "Very easy" due to Rational Rose itself, which did not synchronise the changes made to the *Class* diagram with the *Statechart* diagram. The mistakes were not obvious until they run the model in B tools. In addition, one respondent found that

the multiplicity of associations had to be given more thought during the modelling. This was because unsuitable multiplicity could violate the invariants of the B model even though the multiplicity seemed to make sense in the diagrams. The remaining two respondents thought it was "Difficult" to make mistakes because even if they did, the mistakes could easily be identified and corrected.

In contrast, eight of the respondents believed that it was "Easy" and "Very easy" to make mistakes when defining formal semantics using the B syntax. Since the semantics had to be specified literally through typing, mistakes such as wrong variables names, data types, inappropriate use of clauses and typing errors could easily be made. Moreover, the syntax checking had to be done manually as there was no such facility in Rational Rose. Having the semantics scattered around different parts of the models made the task more troublesome, as the semantics could not be viewed easily at once. Any mistakes in applying the B syntax could only be realised when they transformed the UML-B model to a B model and run the model in B tools. Several respondents also highlighted that the mistakes were "Easy" to make due to lack of understanding, documentation and experience on UML-B. In fact, they were also new to B and were novice users of UML, which made them prone to errors.

The remaining two respondents believed that it was "Difficult" to make mistakes due to the formality imposed by the B syntax.

	-2 Very difficult	**-1 Difficult**	**0 Neither difficult nor easy**	**1 Easy**	**2 Very easy**	**Total**
UML-B Diagram	0	2	6	1	1	10
	0%	20%	60%	10%	10%	100%
	20%		60%	20%		100%
Median	0					
UML-B Syntax	0	2	0	6	2	10
	0%	20%	0%	60%	20%	100%
	20%		0%	80%		100%
Median	1					

TABLE 4.4: Distribution of answers for the "Error Proneness" dimension

4.2.5. Progressive Evaluation

The question **(5)** of the survey assessed the ability of UML-B to allow the user to evaluate his or her work in progress at any time. The evaluation process involves the transformation of the UML-B model to the B model using U2B and the execution of B tools. The question required the respondents to indicate whether or not it is possible to stop modelling at any time to check their work so far. The respondents had to state why if it was not possible.

The Table 4.5 below shows the distribution of answers for question **(5)**. It can be seen that majority of the answers were "Yes". The remaining respondents were not sure or thought it was not always possible depending on at what stage they stopped. They believed major elements of the UML-B model needed to be specified correctly before translating the model to the B model. Otherwise, the error messages generated by U2B and B tools would be too intimidating.

	No	Not sure	Yes	Total
UML-B Model & U2B & B tools	1	2	7	10
	10%	20%	70%	100%

TABLE 4.5: Distribution of answers for the "Progressive Evaluation" dimension

4.2.6. Hard Mental Operations

The question **(6)** of the survey assessed the degree of mental processes required for the user to understand the notation and to keep track of what is happening. The question required the respondents to indicate whether or not they found any complex or difficult tasks to work out in their heads when modelling the UML-B model. The respondents had to state what the difficulty was, if any.

The Table 4.6 below shows the distribution of answers for question **(6)**. It can be seen that six respondents stated the answer as "No". One respondent commented that the visual aspect of the UML-B model helped in reducing the hard mental operations, which would exist in the traditional B modelling.

Four respondents found some complex tasks to work out in their heads. Two respondents found that writing correct semantics for the model was hard. One respondent discovered that having semantics in the *Statechart* diagram would make the transformed B model more complex. For instance, the transitions in the *Statechart* diagram were translated as nested conditions in the B model, which could conflict with the already defined conditions. One respondent believed that having to consider and integrate two modelling styles, UML and B, at the same time was indeed a mental burden.

	No	Not sure	Yes	Total
UML-B Model	6	0	4	10
	60%	0%	40%	100%

TABLE 4.6: Distribution of answers for the "Hard Mental Operations" dimension

4.2.7. Consistency

The question **(7)** of the survey assessed whether similar semantics in the notation were presented in a similar syntactic manner. The question required the respondents to indicate whether or not they found any parts in the model that seem to be similar in functionality but the method makes them appear different. The respondents had to state what the parts were, if any.

The Table 4.7 below shows the distribution of answers for question **(7)**. It can be seen that six respondents stated the answer as "No". The remaining respondents were not sure whether or not the parts exist.

	No	Not sure	Yes	Total
UML-B Model	6	4	0	10
	60%	40%	0%	100%

TABLE 4.7: Distribution of answers for the "Consistency" dimension

4.2.8. Hidden Dependencies (ISO: Understandability/Learnability)

The question **(8)** of the survey assessed whether there was any relationship between two parts such that one of them was dependent on the other but the dependency was not fully visible. The question required the respondents to indicate whether or not they found any structure dependencies in the model. If they did, the respondents had to state how visible the structure dependencies and what parts that were involved.

The Table 4.8 below shows the distribution of answers for question **(8)**. It can be seen that four respondents stated the answer as "Yes". Three of these respondents found that those structure dependencies were visible in both parts. One of them commented that the dependencies were visible only in one part. As pieces of information were scattered around different parts of the UML-B model, the relationship between these parts were not so visible until the model was transformed to a B model by U2B.

The remaining respondents were not sure whether or not the parts exist.

	No	Not sure	Yes	Total
UML-B Model	3	3	4	10
	30%	30%	40%	100%

TABLE 4.8: Distribution of answers for the "Hidden Dependencies" dimension

4.2.9. Secondary Notation

The question **(9)** of the survey assessed the ability of UML-B to allow the user to provide supporting information to the model by using notation other than the official semantics. As the UML-B model resides in Rational Rose, the assessment particularly concerned the ability of the application to support the above user's activity. The question required the respondents to indicate whether or not they could make notes or convey extra information beyond the model to themselves. The respondents had to state the possible actions, if any.

The Table 4.9 below shows the distribution of answers for question **(9)**. It can be seen that all respondents stated the answer as "Yes". The respondents found that the notes and the documentation facility in Rational Rose were very useful for this purpose. Three respondents mentioned that the notes would be very helpful for formal models.

	No	Not sure	Yes	Total
UML-B Model & Rose	0	0	10	10
	0%	0%	100%	100%

TABLE 4.9: Distribution of answers for the "Secondary Notation" dimension

4.2.10. Role Expressiveness

The question **(10)** of the survey assessed whether the purpose of each component in the model was obvious and the user could directly imply how it related to the whole model. The question required the respondents to indicate how easy to determine what each diagram and syntax was for in the UML-B model as a whole. In addition, the question also asked whether the respondents included any component in the model without exactly knowing its purpose.

The Table 4.10 below shows the distribution of answers for question **(10)**. For the diagrams, it can be seen that five respondents considered the task as "Easy" and "Very easy". This causes the median to be "Easy". These respondents found that the concepts of UML were easy to grasp. There were a lot of resources on UML that they could refer. Once the concepts were known, they could easily differentiate the role of each part of the diagrams.

On the other hand, four respondents considered the task as "Neither difficult nor easy". These respondents did not really understand why they needed to have the *Statechart* diagrams, as they believed they could simply use the *Class* diagrams to specify the behaviours. In addition, they were also quite confused about the roles of *Precondition* and *Post-condition* in the diagrams. As far as the UML-B modelling was concerned, they believed they could merely use the *Semantics*.

For the B syntax, four respondents considered the task as "Neither difficult nor easy". Despite being taught on B, these respondents faced some difficulties in dealing with the B syntax. Three respondents considered the task as "Difficult" due to the same reason. They believed more experience and time were required to fully understand the roles of B syntax in the UML-B model and how they could work together. Besides, they believed more comprehensive documentation should be available to support them during the process.

Three respondents found the task as "Easy" particularly after the major parts of the model had been illustrated using the diagrams. The structure of the diagrams somehow helped them in determining the roles of the B syntax.
Two respondents found that there were parts that they simply included without knowing the purpose; the *Statechart* diagram and *Post-condition*.

	-2 Very difficult	**-1 Difficult**	**0 Neither difficult nor easy**	**1 Easy**	**2 Very easy**	**Total**
UML-B Diagram	0	1	4	4	1	10
	0%	10%	40%	40%	10%	100%
	10%		40%	50%		100%
Median	1					
UML-B Syntax	0	3	4	3	0	10
	0%	30%	40%	30%	0%	100%
	30%		40%	30%		100%
Median	0					

TABLE 4.10: Distribution of answers for the "Role Expressiveness" dimension

4.2.11. Closeness of Mapping

The question **(11)** of the survey assessed the mapping between the notation used in UML-B and the problem domain. The question required the respondents to indicate how well UML-B allowed them to describe their problem accurately and completely as what they intended.

The Table 4.11 below shows the distribution of answers for question **(11)**. It can be seen that six respondents regarded the mapping as "Good" and "Very good". This causes the median to be "good". Three of the respondents believed that the

mapping was achieved easily because of UML and its object-oriented concept. Two respondents commented that UML and Rational Rose had guided them through the modelling process in a logical way, which helped in ensuring a complete model to be developed. They started the modelling with the UML diagrams, which provided the overview of the whole system. The overview later leaded them to specify the system behaviours in more detail and systematically. One respondent believed that UML-B and U2B were useful for the development of a B model, which would be different if the B model was developed from scratch.

Four respondents considered the mapping was "Neither bad nor good". The respondents found several occasions where they wanted to add certain features, which seemed to be logical in UML, but did not work well in UML-B. In turn, they had to change slightly the way they normally did in UML in order to accommodate the UML-B modelling style.

	-2 Very bad	**-1 Bad**	**0 Neither bad nor good**	**1 Good**	**2 Very good**	**Total**
UML-B Model & U2B	0	0	4	6	0	10
	0%	0%	40%	60%	0%	100%
	0%		40%	60%		100%
Median	1					

TABLE 4.11: Distribution of answers for the "Closeness of Mapping" dimension

4.2.12. Provisionality

The question **(12)** of the survey assessed the flexibility of UML-B. The question required the respondents to indicate how well the method allowed them to play around with the model without being sure what the effect would be. The respondents were required to state which parts of the method that allowed or prevented them to do so.

The Table 4.12 below shows the distribution of answers for question **(12)**. It can be seen that five respondents commented that the notation was not good enough for them to play around with the model. These respondents agreed that they could

make any changes to the UML-B model to test any new ideas. However, their main concern was that they needed to transform the UML-B model to a B model each time they made changes so that they could test the model using B tools. Otherwise, there was no way they could be sure whether or not the ideas were correct, as Rational Rose did not support any syntax or model checking.

Four respondents found that they could easily play around with the model. These respondents believed that the concepts of UML in the UML-B model had made the process easier. Although they admitted that they needed to transform the UML-B model to a B model in order to test the effects, they did not found it as a burden. Being able to test the model using B tools was regarded as one of the method's strengths.

	-2 Very bad	**-1 Bad**	**0 Neither bad nor good**	**1 Good**	**2 Very good**	**Total**
UML-B Model & Rose & U2B	2	3	1	3	1	10
	20%	30%	10%	30%	10%	100%
	50%		10%	40%		100%
Median	-0.5					

TABLE 4.12: Distribution of answers for the "Provisionality" dimension

4.2.13. Premature Commitment

The question **(13)** of the survey assessed whether the notation used in UML-B enforced the user to make decisions prior to modelling or there was any task ordering constraints. The question required the respondents to indicate whether or not they could go about any task in any order they liked.

The Table 4.13 below shows the distribution of answers for question **(13)**. It can be seen that nine respondents commented that there was no task ordering constraints. They generally believed that they could start modelling as they liked. However, they found it was more logical to start with the diagrams before specifying the semantics for the operations using the B syntax.

	No	Not sure	Yes	Total
UML-B Method	0	1	9	10
	0%	10%	90%	100%

TABLE 4.13: Distribution of answers for the "Premature Commitment" dimension

4.2.14. Abstraction Gradient

The question **(14)** of the survey assessed whether the notation used in UML-B enforces any level of grouping mechanism. The question required the respondents to indicate whether the method insisted they start modelling task by defining or grouping things before they could do anything else.

The Table 4.14 below shows the distribution of answers for question **(14)**. It can be seen that six respondents commented that they did not think the method insisted them to define or group things when they started the modelling. They generally found the process was natural. They would define or group things whenever required.

On the other hand, three respondents found that they had to define the classes needed and group the attributes and operations according to those classes, before they could proceed.

	No	Not sure	Yes	Total
UML-B Method	6	1	3	10
	60%	10%	30%	100%

TABLE 4.14: Distribution of answers for the "Abstraction Gradient" dimension

4.2.15. Learnability of UML-B

The question **(15)** of the survey assessed the learnability of UML-B. The question required the respondents to indicate how easy to learn UML-B compared to traditional B. The respondents were also required to indicate any particular parts

of the method that were particularly difficult to learn and understand how they work.

The Table 4.15 below shows the distribution of answers for question **(15)**. It can be seen that four respondents found that UML-B was "Difficult" and "Very difficult" to learn. This was because the respondents had to integrate two concepts of modelling, that is, UML and B. As U2B transformed the UML-B model to B model automatically, they had to understand how the transformation was done. They had to know what effects that the generated B model would have for each feature that they added on the UML-B model. Familiarity with tools such as Rational Rose was also believed to play a role on the method's learnability.

Three respondents thought the method was "Neither difficult nor easy" to learn. Similarly, three respondents commented the method as "Easy" and "Very easy" to learn. These respondents believed that learning the method was easy because of the UML diagrams. However, they would foresee that learning the method would become difficult if they had not been taught on UML and B.

	-2 Very difficult	**-1 Difficult**	**0 Neither difficult nor easy**	**1 Easy**	**2 Very easy**	**Total**
UML-B vs B	3	1	3	2	1	10
	30%	10%	30%	20%	10%	100%
	40%		30%	30%		100%
Median	0					

TABLE 4.15: Distribution of answers for the learnability of UML-B

4.2.16. Learnability and Utility of U2B

The question **(16)** of the survey assessed the learnability of U2B that accompanies UML-B. U2B is a tool that transforms a UML-B model to a B model so that it could be verified by B tools. The question required the respondents to indicate how easy to learn and use U2B. The respondents were also required to indicate whether the tool had met its purpose and their expectation.

The Table 4.16 below shows the distribution of answers for question **(16)**. It can be seen that all respondents found that U2B tool was "Easy" and "Very easy" to learn and use. Even though the tool lacks documentation on how to use, these respondents found the process was straightforward. Simply following a short instruction and clicking a button, the UML-B model could be transformed to a B model.

Five respondents agreed that the tool had successfully met its purpose and their expectation. The tool had helped them in developing a correct model. These respondents would consider using UML-B to generate a B model rather than developing a B model from scratch. However, some of them admitted that using the tool for the first time was quite daunting as the tool generated a vast amount of syntax. They therefore had to understand why and how the transformation was done. Four respondents thought the tool had helped them "A little" as it only transformed the UML-B to a B model. Much of the difficult tasks such as specifying correct semantics and verifying the model still needed to be done by them.

	-2 Very difficult	**-1 Difficult**	**0 Neither difficult nor easy**	**1 Easy**	**2 Very easy**	**Total**
U2B	0	0	0	5	5	10
	0%	0%	0%	50%	50%	100%
	0%		0%	100%		100%
Median	1.5					

	-2 No	**-1 Not Sure**	**0 Yes, a little**	**1 Yes**	**2 Yes, a lot**	**Total**
U2B	0	1	4	5	0	10
	0%	10%	40%	50%	0%	100%
	10%		40%	50%		100%
Median	0.5					

TABLE 4.16: Distribution of answers for the learnability and utility of U2B

4.2.17. Usefulness of Documentation

The question **(17)** of the survey assessed the usefulness of the available manual and documentation on UML-B.

The Table 4.17 below shows the distribution of answers for question **(17)**. It can be seen that five respondents found that UML-B documentation was "Neither useful nor useless". These respondents generally found that the documentation was quite complicated to understand. In fact, they found that the presentation slides used during the lecture was more useful than the documentation. They used the slides extensively during the model development.

Four respondents commented that the documentation was "Useless". These respondents found that the documentation merely discussed the theory underlying the method rather than specific examples on how to build a UML-B model step-by-step from scratch. They faced some difficulties in understanding the practical aspect of the method such as why certain things should be done in certain ways. They would expect more comprehensive documentation on the method.

One respondent found that the documentation was useful.

	-2 Very useless	**-1 Useless**	**0 Neither useless nor useful**	**1 Useful**	**2 Very useful**	**Total**
UML-B & Doc	0	4	5	1	0	10
	0%	40%	50%	10%	0%	100%
	40%		50%	10%		100%
Median	0					

TABLE 4.17: Distribution of answers for the usefulness of documentation on UML-B

4.2.18. Accessibility of UML-B

The question **(18)** of the survey assessed the accessibility of UML-B. In particular, the question required the respondents to indicate how easy to become familiar with the method and to be able to use it in their task efficiently without referring to the documentation.

The Table 4.18 below shows the distribution of answers for question **(18)**. It can be seen that four respondents found that it was "Easy" and "Very easy" to become familiar with the method. Once they were clear on how to use the notation correctly and had some practice in using it, the task was pretty straightforward

where the documentation could be neglected. However, they admitted that the difficult part was to understand how the notation and the transformation worked as a whole.

Four respondents felt that the task was "Difficult" and "Very difficult" because the method integrates both UML and B. They found that learning these two notations particularly the B syntax took much of their time. Moreover, they had to learn how the two notations should be integrated in the UML-B model. They found that using the method was easy but mastering it was quite difficult.

Two respondents commented the task as "Neither difficult nor easy". There were some parts such as the *Statechart* diagram and *Associations* that required them to refer to the documentation quite often.

	-2 Very difficult	**-1 Difficult**	**0 Neither difficult nor easy**	**1 Easy**	**2 Very easy**	**Total**
UML-B	1	3	2	2	2	10
	10%	30%	20%	20%	20%	100%
	40%		20%	40%		100%
Median	0					

TABLE 4.18: Distribution of answers for the accessibility of UML-B

4.2.19. Operability and Attractiveness of UML-B

The question **(19)** of the survey assessed the operability of UML-B. In particular, the question required the respondents to indicate how easy to do modelling using UML-B compared to traditional B. The respondents were also required to indicate their choice in modelling, that is, which method that they would prefer to use in modelling.

The Table 4.19 below shows the distribution of answers for question **(19)**. It can be seen that four respondents found that it was "Easy" and "Very easy" to model a system using UML-B compared to traditional B. This was because the main elements of the model could be illustrated graphically using the UML diagrams. The diagrams made the process of specifying semantics for the model more

obvious. Besides, they found that much of the effort and trouble in modelling a B model were reduced through automatic transformation provided by U2B. The tool was seen as capable of preventing more errors to be made on the model.

Four respondents regarded the task was "Difficult" and "Very difficult" because they had to integrate both styles of modelling, UML and B, at the same time. Different ways of specifying the semantics had caused some confusion to these respondents. In addition, the lack of training and comprehensive documentation on UML-B was also a factor that made it difficult.

Six respondents preferred UML-B to traditional B. These respondents believed that UML-B would be useful and easier to use if they were given more time and exposure to the method. They could see the potential of the method as it is much more closer to the realism.

	-2 Very difficult	**-1 Difficult**	**0 Neither difficult nor easy**	**1 Easy**	**2 Very easy**	**Total**
UML-B vs B	1	3	2	4	0	10
	10%	30%	20%	40%	0%	100%
	40%		20%	40%		100%
Median	0					

	UML-B	**B**	**Both**	**Total**
UML-B vs B	6	2	2	10
	60%	20%	20%	100%

TABLE 4.19: Distribution of answers for the operability and attractiveness of UML-B

4.2.20. Further Improvement

The question **(20)** of the survey provided the respondents an opportunity to raise any issue of using UML-B and U2B. The respondents were also allowed to suggest any possible improvement that could be made on the method and its tools.

Below are some of the issues and areas for improvement highlighted by the respondents:

- Provide syntax checking at early stage (UML-B model rather than B model)
- Provide dropdown lists for B syntax to avoid typing errors
- Provide automatic changes in the respective parts of the model
- Provide a more functional and user-friendly interface for U2B
- Provide comprehensive documentation

4.2.21. Other Findings

The respondents were among the subjects who were involved in the controlled experiment conducted previously. An informal observation had been made where the performance of the respondents when interpreting a UML-B model had been compared with their perception when developing the model. From the Table 4.20 below, it can be seen that seven respondents perceived UML-B as better than B in the survey. This includes four respondents who performed better using B in the experiment. On the other hand, two respondents who preferred B for modelling performed better on UML-B in the experiment. Only four respondents had been found to be consistent across the two studies, three for UML-B and one for B. This seems to suggest that there is a difference between model interpretation and creation tasks using the method. This may be because the models used in the experiment were on papers whereas the survey required the model to be developed using the UML-B tools. Perhaps, the respondents could make more senses of UML-B when using it online.

Respondent	Experiment (Better Performance)	Survey (Better Perception)
R1	U	B
R2	B	U
R3	B	U
R4	U	U
R5	B	U
R6	U	B
R7	U	U
R8	B	U
R9	U	U
R10	B	B

TABLE 4.20: Performance and Perception of UML-B

4.3. Analysis

The survey adopted the grounded theory approach for the data analysis. Besides capturing the experience of using UML-B, the survey aimed to formulate tentative theories of usability of such integrated methods in general. The theory in the approach denotes a set of discrete categories that are systematically connected through statements of relationship. The categories in essence are abstract concepts that describe the phenomenon under study whereas the statements of relationship are the interrelated properties of those categories.

Employing the grounded theory approach entails a certain amount of coding and analysis. The first one was *open coding* where the responses were examined for objects of interest based on the stated research questions. The technique used was *microanalysis* (Strauss et al., 1998). The analysis focused on the identification of major themes or categories and how often they emerged in the data under varying conditions. The idea was to form a theoretical framework, thus the analysis involved the formulation of general categories rather than specific to any individual cases. For example, issues of using Rational Rose and running U2B were conceptualised as *Availability and usefulness of supporting tools*. The analysis did not intend to specifically delineate every single limitation of the tools. Rather, the objective was to identify and propose a set of categories that can be used as a basis for examining the usability of other similar methods in future.

After completing *open coding*, *axial coding* process was conducted. *Axial coding* involves moving to a higher level of abstraction by identifying relationships between categories based on their properties. This forms the basis for the theory construction. The properties for the categories were derived by having queries such as *what*, *why*, *how* and *when* during the analysis process. For example, respondents mentioned the issue of learning UML and B several times in their answers. Therefore, *Learnability of notations and tools* was recognised as one of the categories. On the other hand, it is necessary to know *what* aspect of the notations and their tools that was easy and difficult to learn, *when* and *why* they happened, in order to understand the phenomenon. To answer the queries,

evidence was obtained and accumulated from various parts of the questionnaire. This included both the quantitative (ordinal scale) and qualitative (subjective) data. As an exploratory study, the analysis mainly focused on the qualitative answers as they provided a richer explanation of the phenomenon. The qualitative answers were particularly useful when the quantitative answers were inconsistent where some respondents gave positive answers and some gave negative answers with no clear majority or consistent trends. The use of CD framework and ISO's usability criteria that shapes the dimensions of usability investigation facilitated the identification of the categories and properties.

The following sub-sections list the categories and elaborate their properties. The properties were grouped into categories based on the respondents' qualitative and quantitative answers. The properties (reasoning based on CD and ISO's usability criteria) that support the statements are stated in the parentheses in the paragraphs, which link to the actual evidence described in the previous section.

4.3.1. Category 1: Model Structure and Organisation

The UML portion of UML-B allows the system properties and behaviours to be illustrated using the *Class* and *Statechart* diagrams. Each diagram represents the system from a specific perspective. For example, the *Class* diagram shows the attributes and relationships between entities in the system while the *Statechart* diagram delineates the states and transitions involved in the system operations. In modelling a UML-B model, the users employ the diagrams to illustrate the system properties from these perspectives.

The diagrams are equipped with formal semantics where the characteristics and behaviours of the systems are specified more precisely. Formal semantics in the form of B syntax are added at different parts of the diagrams so that they can be transformed to a B model. For example, the global variables and invariants are placed at the *Class* diagram level while the conditions and effects of the behaviours are placed at the *Statechart* diagram level. Despite being scattered at several parts of the model, the method has the ability to transform the diagrams and consolidate the semantics as a single B model through its tool, namely U2B.

Despite being logical, having the formal semantics at different parts of the model causes an accessibility issue to the users. They need to switch around different parts of the model to specify the formal semantics. Rational Rose supports the display of multiple windows at one time. However, having to deal with several displayed windows simultaneously in Rational Rose seems to be a problem (Property: "Visibility and Juxtaposibility" dimension). The users have to view not only the windows that display the *Class* and *Statechart* diagrams but also the pop-up windows that carry the semantics for each of the diagrams. In fact, some of these windows have to be on top of each other due to limited screen space. This leads the users to overlook certain aspects of the model and prone to errors (Property: "Error Proneness" dimension). The users can view and subsequently check the model using B tools by translating it to a B model using U2B at any modelling stage they like (Property: "Progressive Evaluation" dimension). However, having to transform the model particularly during formulating and synthesising ideas has been regarded as a 'noise'. In addition, model transformation at early stages where many aspects have yet to be given careful thought will generate error messages in B tools. Starting modelling with many generated errors can be a daunting experience especially to new users.

This finding supports the comment obtained from the controlled experiment where the UML-B model had been regarded as 'messy'. The 'messiness' is not only caused by the scattered information but also the display of multiple windows at a time. The structure of the model does affect its accessibility for both model reading and development, even on the computer screen. The cognitive psychology theory that underpins this phenomenon is that humans have a limited amount of information that can be processed at one time. The way material is organised and presented has an effect (Chandler et al., 1992). When the related information is separated from each other on the page or screen, users have to use cognitive resources to search and integrate it. Users are less likely to be able to hold the separated information in working memory at the same time especially if the information has a high intrinsic cognitive load (Sweller et al., 1994). In general, formal notation such as B syntax is high in intrinsic cognitive load because it involves concurrent interactions between its syntactical and semantic characteristics.

As a UML-B model always involves the use of more than one UML diagram that carries the respective B syntax, the issue of scattered information is seen as unavoidable. However, the effect of split-attention can be reduced if the modelling tool allows the switching and viewing of different parts of the model more conveniently and less distracting.

4.3.2. Category 2: Availability and Usefulness of Supporting Tools

Rational Rose and U2B are the main supporting tools in UML-B. These tools have been useful in some aspects (Property: "Consistency" dimension; "Secondary Notation" dimension; "Learnability and Utility of U2B"). On the other hand, there are also several user-friendliness issues discovered by the users. For example, Rational Rose does not support some changes automatically, which causes the modification process to be unnecessarily tedious (Property: "Viscosity" dimension). If a variable name is changed in the *Class* diagram, the change is not reflected in other parts such as in the *Statechart* diagram or in the semantics where the variable name is used. A similar situation applies to variable deletion. Thus, the changes have to be done manually by visiting the respective parts of the model.

U2B in general has received a fairly good acceptance among the users. This is due to its obvious role, that is, to transform a UML-B model to a B model. By executing several simple steps, the users can generate a B model and execute the verification task using B tools (Property: "Progressive Evaluation" dimension). This is the reason why the tool is seen as easy to learn and use (Property: "Learnability and Utility of U2B"). The automatic transformation has alleviated some pains that would occur when modelling a B model from scratch. At the very least, it provides basic structures for the B model where the users could extend further by adding more details. The simplicity of U2B however has made the verification task remains in B tools. No matter how simple, U2B or even Rational Rose does not support any checking in any way. This means the users have to transform the UML-B model to a B model and run it in B tools each time they change ideas even if it involves only a minor change. Otherwise, there is no way they could be sure whether or not the changes are acceptable. The generated B

model will contain numerous types of errors from the simplest to the complex ones, which can only be realised during model verification using B tools. Because of this reason, the users feel that the method is not good enough for playing around with ideas (Property: "Provisionality" dimension). Some simple checking such as unused variables and typing errors of B syntax at the modelling and transformation levels would be useful to the users. This can act as the front line checking, which eliminates trivial errors before pursuing more extensive verification in B tools. Rather than introducing all types of errors at once, evolutionary phases of checking could make the verification task less daunting and troublesome to the users. As the tool lacks of these elements, it does not fully meet the users' expectation (Property: "Learnability and Utility of U2B").

This finding supports the comment obtained from the controlled experiment where several subjects in the experiment believed that the method is useful only with good tool support. Although the necessary tools are available, there are several aspects that should be improved in order to increase their utility (Property: "Future Improvement"). Perhaps a more seamless modelling environment should be created so that users do not have to perform several individual and intricate steps during modelling.

4.3.3. Category 3: Learnability of Notations and Tools

The successful use of UML-B method relies on the fact that users have to be familiar with UML and B. Otherwise, the integration of both notations could not be understood or valued. It has been found that it is difficult if not impossible to obtain the understanding of the notations used in both UML and B at the same time (Property: "Learnability of UML-B"). Even though the users have been exposed to UML and B for some time, some mental burden still occurs during the process (Property: "Hard Mental Operations" dimension). Having to think, integrate and harmonise two styles of modelling from two different methods seems to be problematic.

The model transformation provided by U2B also requires some learning (Property: "Learnability of UML-B"). A UML-B model in essence carries two

types of semantics; explicit B syntax specified by the users in the UML diagrams that U2B transforms as it is in the B model, and implicit B syntax that U2B implies and generates automatically from the diagrams. For example, behaviours of the operations have to be specified by the users using the B syntax in the UML diagrams whereas classes and associations in the diagrams are translated automatically as the respective sets and variables in the B model. The users have to understand these transformations and why they are accomplished in such ways (Property: "Learnability and Utility of U2B"; "Hidden Dependencies" dimension) as it affects the way they should do the modelling (Property: "Closeness of Mapping"). Moreover, learning of how to do modelling in Rational Rose is also required (Property: "Learnability of UML-B").

Modelling the UML diagrams is regarded as quite straightforward (Property: "Role Expressiveness-Diagram" dimension; "Error Proneness-Diagram" dimension), which eases the process of describing what is intended (Property: "Diffuseness" dimension; "Closeness of Mapping" dimension). Despite the fact that B modelling imposes some task ordering and requires users to define and group things beforehand, the diagrams have somehow diluted the effects (Property: "Premature Commitment" dimension; "Abstraction Gradient" dimension). Perhaps these factors help to explain why a UML-B model is seen as more approachable than a B model and thus, UML-B is preferred for formal modelling (Property: "Operability and Attractiveness of UML-B").

On the other hand, specifying the UML diagrams with the correct formal semantics is perceived as hard and error-prone (Property: "Error Proneness-Syntax" dimension; "Hard Mental Operations" dimension). Shallow understanding of how the formal semantics should work with the UML diagrams, lack of comprehensive documentation on the method (Property: "Usefulness of Documentation") and the need to grasp the underlying principles of the participated methods and tools mentioned above have downgraded the operability of the method (Property: "Operability and Attractiveness of UML-B"). To attract new users to the method, a more comprehensive documentation should be readily available (Property: "Future Improvement"). The documentation should cover more on the practical aspect of the method and its tools (Property: "Usefulness of

documentation"), rather than just theory. Currently, the available documentation on the method is not helping the users much in this aspect (Property: "Accessibility of UML-B")

4.3.4. Category 4: Functionality of Notations

Rational Rose provides specification windows in each diagram for specifying the semantics. There are two types of diagrams involved in UML-B, thus the users are provided with two types of specification windows. One is in the *Class* diagram and the other is in the *Statechart* diagram. Regardless the location, U2B is able to extract the semantics and treat them accordingly as a B model.

The semantics in the *Statechart* diagram are transformed as a nested condition under the primary condition, which is obtained from the *Class* diagram. In many cases, the semantics of the *Statechart* diagram can also be placed directly in the specification windows of the *Class* diagram. If the users know what the states and transitions involved in the operations, they can specify it literally as a series of conditions in the specification windows of the *Class* diagram. Despite providing an alternative in modelling, the flexibility somehow has made the role of the semantics in the *Statechart* diagram or even the *Statechart* diagram unclear to some users (Property: "Role Expressiveness-Diagram" dimension; "Role Expressiveness-Syntax" dimension). The users seem to prefer specifying the whole semantics in the *Class* diagram, as it is more obvious and straightforward. It could also reduce the mental burden of having to work with two different diagrams at the same time (Property: "Visibility and Juxtaposibility" dimension; "Hard Mental Operations" dimension). Moreover, the generated nested conditions from the *Statechart* diagram tend to complicate the B model. As the end product that actually matters is the transformed B model, the users prefer to have a simple and quick solution to achieve it.

More clear roles and boundaries should be set between the formal semantics of the *Class* diagram and the *Statechart* diagram. The explanation on the roles and responsibilities of each part of the diagrams and semantics should be stated succinctly in the documentation, which the method is currently lacking (Property:

"Usefulness of documentation"). It may be better if some principles and controls can be placed on how a UML-B model should be modelled. Although it may reduce the flexibility in modelling, it can at least guide the users based on what should and should not be done. It can also avoid redundancy. This is particularly true for new users who mainly have no idea on how to start and pursue the modelling. Besides, the transformation of formal semantics from the *Statechart* diagram to a B model can be smoothed further so that no unnecessary complication is introduced to users.

4.4. Discussion

There are two main outcomes of the survey, namely theory generation and proposal of a design profile. The following sub-sections explain the theories and the design profile respectively.

4.4.1. Theory Generation

The data from the survey suggest that UML-B is appealing to users who opt into B modelling while yet prefer working with the standard development style of UML. This is particularly true when users are familiar with UML and have the capacity to appreciate what formal notations such as B could offer. The graphical modelling environment alleviates the pain of developing a formal model from scratch by stimulating the formulation of idea through the use of visual objects at the abstraction level. Some comments from the respondents that support this claim include:

Respondent 1:

"UML–B is based on UML modelling. Since the modelling in UML is very precise and robust, UML-B modelling is definitely easier."

"When I developed my model in UML-B, I could switch to B tool instantly every time I added a method to ensure that I haven't produced an error. This is good and easy."

Respondent 3:

"I'm familiar with UML and object-oriented technology. It seems to me that UML-B has the objected-oriented style. The diagrams make it easier to build up a model. This is a power of UML-B as you can think graphically and model it semantically."

Respondent 5:

"Modelling with UML-B is much closer to the realism as compared to B modelling".

Respondent 7:

"I had very low expectations about UML-B before I used it and wasn't very satisfied at first because it was hard to get started due to the lack of documentation. But after a few hours, I really enjoyed using it and I can imagine it to be useful in industry. It created a lot of code even though only a little bit of B notation was added to the model. This can save a lot of time and prevent many errors. If I had to write another B machine, I would consider using UML-B."

Respondent 8:

"UMLB is much easier to model than just in B because you have the visual model, which makes it easier to think through different tasks".

Respondent 10:

"UML-B model use both graphical and semantics for modelling thus easier to work with. UML-B is a nice tool to describe a model because you can start with a visual model (i.e. the class diagram) and then B lets you add more detail."

On the other hand, users are faced with the challenge of having to grasp the underlying principles of each individual notation as well as to understand how both notations work together to achieve the integration objectives. Each notation's roles and functionality at different parts of a model should be understood, which can easily be achieved only if the distinction between them is clear-cut. Users are also required to learn and become familiar with the individual tools that accompany each notation, which in general should provide the necessary support. Some comments underlying this claim are as follows:

Respondent 1:

"The main reason of difficulty is the lack of experience and documentation concerning UML-B. In addition, a good knowledge of both UML and B-Method is required in order to formulate the semantics. It was necessary to first gain familiarity with UML prior to building the model."

"I have never used Rational Rose before. So, I had to learn how to use it then learn how to use UML-B."

Respondent 2:

"It is difficult to think in UML and B at the same time. I had to gain familiarity with UML diagrams before building the model. Once the code is generated, we need to read the code entirely to understand what it does and how it functions to be able to make changes in the UML-B diagrams."

"If no prior experience with UML is acquired, it is difficult to determine the meaning of the diagrams and components."

Respondent 3:

"The lack of UML-B documentation and examples made it difficult to build the model."

"The tool surely generates the B-Model of the system. However, in order to test the model it is necessary to understand how it is written and how it is generated."

"If I had prior experience with UML, gaining familiarity with UML-B wouldn't be a hard task. However, much time was spent on learning UML and it was necessary to refer to the documentation. And, more experience and time are required in B notations to be comfortable with UML-B."

Respondent 4:

"Once it is clear where to write things, and how the notation works, it is very easy. The only thing is that we have to know UML and B before being able to learn UML-B."

"I think it would be good if there were a few simple examples, and a tutorial that walks you through those examples step by step. The current manual seems to more talk about the theory, not so much about where to put which part of the notation."

Respondent 6:

"I haven't quite worked out what the statechart is for, so I'm confused!"

"There are many ways to do a particular thing this makes it a bit confusing (e.g. specifying an operation in the state chart diagram and class diagram and also some part through the transitions). All of them have a facility to specify preconditions and post conditions. What is the difference between semantics and post conditions?"

Respondent 8:

"If the user is familiar with B and with UML, it will not be that difficult to build a UML-B model."

"There is strong support and help in UML modelling. Since we are already familiar with the UML model, it is easy to associate any diagram."

"B is not a published and robust language. There is no online help that can guide while modelling. Hence it is difficult to understand a B notation unless a person has read the subject in depth from the available resources."

Respondent 9:

"UML-B is based on B. B methods are based on systematic semantics and it took a considerable amount of time to understand the B concepts. Hence, to cover all topics of B in UML-B, it becomes necessary to refer to the document to find out how the modelling can be done corresponding to B. (e.g. Specification of general Quantifiers, Functions Relations, Set Operation, Refinement…inclusion, etc)."

Based on these findings, the survey generates the following tentative theories of the usability of integrated methods that combine semi-formal and formal notations. The categories that contribute to the formulation of the theories are stated in the parentheses:

Theory 1: The integration of semi-formal and formal notations requires the understanding of principles and roles of both notations as well as the rules of the integration. The principles, roles and rules ought to be obvious to users (**Category 3** and **4**).

Theory 2: The integration of semi-formal and formal notations requires strong support from the environment. Supporting tools and comprehensive documentation should be not only available but also useful, easy-to-learn and easy-to-use (**Category 1**, **2** and **3**).

Unlike the other categories, **Category 1: Model Structure and Organisation** is not explicitly stated in the theories although it is included. It is indirectly implied in **Theory 2** with a similar effect as **Category 2: Availability and Usefulness of Supporting Tools**. This is because the incident may depend on the environment by which the method is supported (Rational Rose). Perhaps only the current environment has the problem of managing scattered information and multiple windows. As the data are quite limited, more observation is required on this aspect particularly under different environments.

4.4.2. Design Profile

In terms of the CD framework, goals for designing integrated methods such as UML-B were identified. The design goals were proposed based on the nature of semi-formal and formal notations, and the motivation behind the integration. The individual notations (semi-formal and formal) have their own strengths and weaknesses, which are enhanced through the integration effort. In addition, the design goals were also based on the common types of user activity involved in using such methods. In general, there are two major user activities: *Exploratory*

Design where users use such methods to create a new model, and *Modification* where users use the methods to make changes and enhancements to an existing model.

The Table 4.21 below illustrates the recommended CD profile for designing methods that combine semi-formal and formal notations. The profile proposes the desired level for each dimension that integrated methods and their notations (combination of semi-formal and formal) should aim to achieve. The *High* and *Low* indicate whether the dimension should be increased or reduced respectively, when such methods are designed. For example, method designers are recommended to aim at increasing *Progressive Evaluation* and reducing *Hidden Dependencies*. The *Moderate* indicates that although the dimension is desired at a certain level (*High* or *Low*), it may be traded-off to suit more important dimensions or the two user activities. For instance, *Secondary Notation* is very useful for *Modification* activity as it provides users with additional informal information. It thus may be needed (*High*) to improve the model comprehensibility, especially for formal mathematical models (See Section **4.2.9. Secondary Notation**). However, *Secondary Notation* may cause *Exploratory Design* activity a bit cumbersome, as users are obliged to provide informal information about the elements in the model besides the official notation. Moreover, the two user activities require a model to be less resistant to change (*Low Viscosity*). By having *Secondary Notation*, any alterations to the model can be painful as the changes are also required for the additional information (See Section **4.2.2. Viscosity**). Therefore, *Secondary Notation* may be traded-off (*Moderate* instead of *High*) for achieving *Low Viscosity* and facilitating the two activities. *Diffuseness* may need to be traded-off (*Moderate* instead of *Low*) for achieving *Low Premature Commitment*. *Premature Commitment* is one dimension that designers may aim to reduce because it can be problematic for both *Exploratory Design* and *Modification* activities. To reduce the need for users to look ahead and make a decision before sufficient information is available during the activities, the notation may need to be verbose (See Section **4.2.3. Diffuseness**; Section **4.2.13. Premature Commitment**). It is up to method designers to decide the best compromise based on their methods' context of use and needs.

There are dimensions that specifically affect a particular notation more than the other. By integrating the notation with the other notation, it is believed that its usability can be improved. The * in the Table 4.21 below indicates the dimension that affects formal notations, which semi-formal notations help to reduce the effect. On the other hand, the ** denotes the dimension that semi-formal notations lack, which formal notations help to overcome. For example, it is generally known that formal notations such as B syntax involve *High Hard Mental Operations*, which causes comprehension difficult (Finney et al., 1996a; Finney, 1996b; Carew et al, 2005). The use of graphical symbols in semi-formal notations, which is more intuitive, with formal notations should be able to reduce the effect (See Section **4.2.6. Hard Mental Operations**). Similarly, semi-formal notations in general lack mechanisms for a systematic *Progressive Evaluation*. By integrating with formal notations, such an operation is possible (See Section **4.2.5. Progressive Evaluation**). Without such interplay between the two types of notations, the integration effort is not worthwhile. After all, the motivation of such integrated methods is to allow one notation's limitations to be compensated by the strengths of the other. The following paragraphs elaborate how both notations co-operate to achieve the desired level for dimensions other than described above:

- *Abstraction Gradient:* Formal notations impose abstractions since users need to define and group elements into logical entities (*High*). Moreover, to reduce *Viscosity*, users may need to introduce abstractions so that any changes required would be easier. By integrating the graphical symbols of semi-formal notations with formal notations may alleviate the effect as the grouping of elements becomes more apparent (*Low*). (See Section **4.2.14. Abstraction Gradient**).

- *Closeness of mapping:* The problem domain mapping is not quite straightforward using formal notations due to the notations' unfamiliar symbols and underlying rules of interpretation (*Low*). The graphical symbols in semi-formal notations may however facilitate the mapping, as they generally resemble objects in the real world (*High*) (See Section **4.2.11. Closeness of Mapping**).

- *Consistency:* The formality in formal notations enforces consistency, which semi-formal notations solely could not be able to assure (*Low*). By having semi-formal notations together with formal notations could enable a consistent graphical formal model to be developed (*High*). (See Section **4.2.7. Consistency**).

- *Diffuseness*: Similar to natural language, the textual aspect of formal notations may cause a description to be verbose. In contrast, the graphical symbols in semi-formal notations could normally carry meanings in simpler forms. By combining textual and graphical symbols may cause the description to be short and precise (See Section **4.2.3. Diffuseness**).

- *Error-proneness:* The unfamiliar mathematical symbols in formal notations normally induce mistakes (*High*). The accessibility of graphical symbols in semi-formal notations may reduce the tendency of making errors (*Low*) (See Section **4.2.4. Error Proneness**).

- *Premature Commitment:* Formal notations normally require users to look ahead in order to get the right abstractions (*High*). By incorporating the graphical symbols of semi-formal notations into formal notations may reduce the effect as they permit the visualisation of possible interacting entities (*Low*) (See Section **4.2.13. Premature Commitment**).

- *Role-expressiveness:* The roles of mathematical symbols in formal notations are not so obvious to many users due to their complex interpretation rules (*Low*). On the other hand, the graphical symbols in semi-formal notations are mainly intuitive. By combining the graphical symbols together with the mathematical symbols may help users to grasp the roles of the latter (*High*) (See Section **4.2.11. Closeness of Mapping**).

The remaining dimensions without * or ** in the Table 4.21 below, involve factors other than the notations used. The dimensions are *Provisionality*, *Hidden Dependencies*, *Secondary Notation*, *Viscosity* and *Visibility/Juxtaposibility*. Based on the findings of the survey, it is believed that the environment in which the notations reside plays a major role for achieving the desired levels for these dimensions. This includes the structure of the model and the tools that support the notations (See Section **4.2.1. Visibility and Juxtaposibility; 4.2.2 Viscosity; 4.2.8. Hidden Dependencies; 4.2.9. Secondary Notation; 4.2.12. Provisionality**). This claim is worth investigating in future.

Dimension	Desired level
Abstraction Gradient	Low*
Closeness of Mapping	High*
Consistency	High**
Diffuseness	Moderate (instead of Low)*
Error-proneness	Low*
Hard Mental Operations	Low*
Hidden Dependencies	Low
Premature Commitment	Low*
Progressive Evaluation	High**
Provisionality	High
Role-expressiveness	High*
Secondary Notation	Moderate (instead of High)
Viscosity	Low
Visibility/Juxtaposibility	High

Note: High – to increase; Low – to reduce; Moderate – possible trade-off; * – Semi-formal notations support formal notations to achieve the desired level (otherwise, the level will be opposite); ** – Formal notations support semi-formal notations to achieve the desired level (otherwise, the level will be opposite).

TABLE 4.21: The proposed Cognitive Dimensions profile for designing integrated methods (combine semi-formal and formal notations)

The tentative theories and the proposed CD profile above may not be conclusive, where they can be validated and refined further in future investigations. However, they can act as the first step in understanding the nature of integrated methods such as UML-B and providing a meaningful guide to better design.

4.5. Threats to Validity

The four kinds of validity that must be protected in empirical studies are discussed below (Cook et al., 1979a; Perry et al., 2000).

4.5.1. Internal Validity

This validity concerns whether there is a clear cause and effect relationship between the dependent and independent variables (Gauch, 2000). It determines whether the usability assessment of UML-B could be influenced by any other factors than the UML-B and its modelling environment.

4.5.1.1. Instrument

One of the major disadvantages of the survey is the inability to correct any misunderstandings or probe into responses once the completed questionnaire has been returned. One of the questions seems to mislead two respondents unintentionally, namely question **(4)**. The question was intended to assess the tendency of the notation to induce mistakes by asking *How easy to make mistake…*. The ordinal scale provided for the answer was from *Very difficult* to *Very easy*. The question in general was straightforward. However, the combination of positive effect, that is *easy*, and the negative behaviour, that is *make mistake*, seemed to cause the respondents to overlook the question's actual intention. The respondents' selection on the ordinal scale contradicted with the subjective comments given. As the comments contained more information that reflected the respondents' actual perception, the necessary adjustment was made on the respondents' selection on the scale based on the information given in the justification.

Several questions required the respondents to give explanation only if one of the selections was selected. For example, question **(5)** required the respondents to provide further information if they selected *No* and simply continue to the next question if otherwise. The intention was to require the respondents to provide

explanation only on the aspects that the study was interested to investigate further. It could also reduce the respondents' time in completing the survey. However, it has been found that it would be better to allow the respondents to provide explanation regardless of the selection. It is believed that trivial information may lead to some other aspects that are still worth considering especially when dealing with qualitative data. Besides, it could also reduce the possibility of respondents to select answers that do not require further explanation merely because they are reluctant to respond more. One respondent was suspected to behave this way in the survey.

Given the small number of issues raised from the survey questions, it can be concluded the questions had little impact on the validity of the survey results. Nevertheless, the questions could still be enhanced in future so that their intention could be more obvious and allow more flexibility.

4.5.1.2. Selection of Respondents

The respondents were students in the university where the research was conducted. Therefore, their answers might have been biased either in positive or negative ways. On the other hand, the respondents were considered as the most appropriate candidates because they were trained on B and UML-B. The knowledge is necessary for using UML-B. In fact, they also had some experience of using UML-B and thus were able to contribute to the survey. Moreover, they were independent users, who had no personal interest with the technologies involved or direct contact with the research. To reduce the threat, the subjects were advised to give opinions and comments as sincerely as possible.

4.5.1.3. Sample Size and Response Rate

The survey questionnaire was distributed to all fourteen Masters students of Software Engineering course at the University of Southampton, who registered for the "Critical System" course in Spring 2006. Thirteen students responded to the survey. Due to some technical problem, only ten responses were considered for the analysis. Although the number was quite small, the response rate of seventy

percent was considered as appropriate for an initial attempt. Moreover, as a qualitative study, the quality of the data is the focus rather than the quantity. Brief identity screening was done on the four students who were not included. No particular pattern was identified that could have potentially biased the results.

4.5.1.4. Diffusion or Imitation of Response

The respondents were in contact with each other. Thus, there was a risk that they shared or influenced each other's comments about using UML-B. This could not be controlled. During the analysis however, no cases where two or more questionnaires had identical answers were found.

4.5.1.5. Non-committal and Inconsistent Responses

Using an odd number of levels for the ordinal scale may have left open the possibility of non-committal responses, which caused the medians to be "Neither ..nor.." or "Not sure". Although such incidents could be seen in the data, they did not happen often.

Some of the qualitative answers were inconsistent where one-half of the respondents gave positive answers and the other half gave negative answers. This had caused the distributions to be bimodal. Under these circumstances, the analysis was mainly based on the qualitative answers from various dimensions for building the theories.

4.5.2. External Validity

This validity refers to the ability to generalise the results and conclusions of a study to other populations and conditions than those used in the study (Gauch, 2000; Yin, 2003).

4.5.2.1. Students as Respondents

The respondents of this survey were students. They may have not represented software developers as they were less experience and perhaps were likely less motivated. However, the respondents were in the final semester of their Master course and had reasonable amount of experience and knowledge of software development. Half of the students had some working experience. They were seen as valid respondents for the survey.

4.5.2.2. Toy Problem

Due to time and resource constraints, the modelling task given to the respondents was not large and may have not been representative of real software systems. However, the task was believed to be sufficient for the respondents to experience modelling a system using UML-B. In fact, the task required the respondents to explore most of the functionality provided by the method.

4.5.3. Construct Validity

This validity concerns the establishment of correct operational measures for the concepts being studied (Yin, 2003). It is necessary to ensure that the dependent variables are valid measures and the measurement process is conducted appropriately.

4.5.3.1. Dependent Variables

The dependent variables of survey were the fourteen dimensions of CD and five usability criteria of ISO. These variables were seen as appropriate for measuring the usability of UML-B because they covered both notational and operational aspects. CD and ISO's usability criteria are products of research. Thus, their validity and appropriateness as a measure of usability has been assessed to some degree.

4.5.3.2. Analysis Process

One person did the data interpretation and analysis. Although this may pose a threat however, it can be regarded as negligible as the person was an independent user of UML-B.

The grounded theory approach encourages the gathering of further data after analysing the first gathered data. Data collection and analysis should be repeated several times so that more incidents are captured and validated until the theory saturates (Strauss et al., 1998). Moreover, the approach normally uses an in-depth interview as the means of data collection so that the data can be rigorously captured. Due to time and resources constraints, the data collection and analysis were conducted only once through a survey. The findings therefore were based on one set of data. The survey however will be repeated in future.

4.5.3.3. Nature of Study

Surveys and qualitative measures by their nature are retrospective. Therefore, there was a risk that the respondents responded based on what they thought they did rather than what they actually did. Advising the respondents to complete the survey questionnaire as soon as they did the modelling task could have reduced this threat, as the respondents still remembered of what they found during the task.

4.5.4. Conclusion Validity

This validity concerns the ability to draw the correct conclusion about relations between the object of study and the outcome of the survey.

4.5.4.1. Heterogeneity of Respondents

The respondents might have different ability and experience. Thus, there was a risk that the results might have been affected by individual differences. This could not be avoided. As a qualitative study, the variation however could provide richer data for the analysis.

4.5.4.2. Familiarity of Respondents

The respondents were taught formally on B for about nine hours and one hour on UML-B. They were assigned a modelling task using UML-B within a month period. The period might have been insufficient for the respondents to fully experience the method. The results may have been different if the respondents were given more time and training. The aim of the survey was to capture the experience of using UML-B from new users' perspective. Therefore, the allocated time frame and training were seen as adequate and realistic for the purpose.

The results may also have been influenced by the respondents' experience of UML, which varied considerably.

4.6. Conclusions and Future Work

This chapter has presented an empirical assessment in the form of a survey conducted on a method that integrates the use of semi-formal and formal notations, namely UML-B. The survey assessed the usability of UML-B and its tools from the perspective of new users for conceptual modelling. The objective of the survey was to gain a novel understanding of the method particularly the psychological and physiological effects that it has on its users. The usability assessment was conducted using the Cognitive Dimensions of Notations (CD) framework in concert with the usability criteria suggested in the International Organization for Standardization (ISO). As the survey attempted to understand the nature of experience of using the method, it employed the qualitative approach of the grounded theory. The approach allows the formulation of theoretical explanation based on the feedback received from the users.

The survey has indicated that the dual characteristics of the method bring several implications to users in both positive and negative ways. Combining semi-formal and formal notations allows the potential of individual notation to be strengthened while each notation's limitations can be compensated by the other. However, the

integration in essence brings the loads of two individual notations, which are actually quite different in some ways. Users are required to grasp the principles of the individual notations as well as the rules of the integration. Users therefore need strong support from the environment to lessen the burden that lies beneath the integration effort. The support involves not only the tools that aid the modelling process but also resources for learning the method.

Some of the findings of the investigation are now being fed into the next generation of UML-B development. The findings can be improved further by extending the survey to a large number of users. This will help in enhancing the current understanding of the method and discovering any other factors that affect its use. The tentative theories and the proposed CD profile of integrated methods (combine semi-formal and formal notations) discussed in this chapter can also be validated and refined further by applying them to examine other similar methods. This allows the derivation of more concrete theories and guidelines that can be used to design and improve the usability of such methods in future.

Chapter 5

Measuring the Comprehensibility of a UML-B model and an Event-B model

Conceptual models, which are developed in the early phases of software development, describe aspects of the physical and social world for the purposes of understanding and communication (Mylopulos, 1992). The models are generally used for developers to reason about a problem domain, for communication between stakeholders and for documenting software requirements for future reference (Kung et al., 1986). These activities are important in any software development, for which effectiveness can be achieved only if the models are readily comprehensible.

Model comprehensibility is defined as how easy or difficult to understand the end product of a modelling process. The comprehensibility of a model can be assessed through empirical studies by considering not only the internal characteristics of the notation used in the model but also the cognitive aspects involved in the comprehension process. Drawing general conclusions from empirical studies is difficult (Basili et al., 1999). Results obtained from one empirical study are insufficient to provide conclusive evidence of a particular phenomenon. Therefore, replications of similar investigation are desired so that the results of several studies can be validated and refined.

This chapter presents another experiment conducted on UML-B, which assessed the comprehensibility of its model. This experiment (Experiment 2) replicated the previous experiment (Experiment 1) described in **Chapter 3** on the new version of UML-B. In comparison with the previous one, the new UML-B contains some adjustments in the modelling environment and profiling. In Experiment 2, a model developed using the new version of UML-B was compared with an equivalent Event-B model. Event-B on the other hand is a formal notation evolved from the classical B (Abrial et al., 1998; 2007). The nature of the notations used in Experiment 2 remains the same as in Experiment 1. In particular, the new UML-B model contains graphical and textual representations whereas the Event-B model is mainly textual.

In Experiment 2, the notion of comprehensibility was extended to include problem domain understanding. The experiment aimed to explore the ability of the UML-B model to sustain and promote model viewers' understanding of the presented problem domain. It focused on the ability of model viewers to use the presented information in novel situations. The distinction between Experiment 1 and Experiment 2 is that the attention shifts from the model to include the viewers' developed mental models. In other words, it measures the deep understanding developed when a person views a model. A UML-B model is comprehensible if it allows viewers to not only recognise the presented information but also to extend the understanding of the presented information in novel situations such as problem solving.

The rationale of this investigation is twofold. First, stakeholders communicate and reason about a problem domain to improve their understanding. Without deep understanding of the problem domain, the proposed solutions may not meet the requirements. Second, stakeholders are skilled human beings who use complex cognitive processing when perceive and understand things. When interpreting a model, it is believed that they do not simply "vacuum" the presented information into their mind. Rather, they actively process the information by selecting only the relevant information, organise the selected information into meaningful mental models and integrate them with other knowledge. Interpreting a model can thus be

seen as knowledge construction where stakeholders actively make sense of a problem domain rather than passively receive the information.

The following section of the chapter provides brief description of the new version of UML-B. Section 5.2 discusses the underlying theoretical background that supports the experiment. Section 5.3 to 5.7 explains the technical aspects of the experiment's preparation and execution. Section 5.8 discusses the results and data analysis. Section 5.9 explains several threats to the validity of the results. Finally, Section 5.10 concludes the chapter with a summary of the main findings and future work.

5.1. UML-B (Previous and Current)

The previous version of UML-B is a specialisation of UML using the profiling extension mechanism included in UML. In particular, it uses a subset of UML features that are useful for translation into B. A translator U2B (Snook et al, 2004b), translates a UML-B model into a B model for verification. As demonstrated in the previous survey in **Chapter 4**, the degree of integration between the supporting tools is poor and unidirectional. Moreover, experience with the previous version indicates that the richness and semantics of UML can be misleading. Users were confused over which features they should use. In fact, experienced UML users claimed that the semantics of UML-B is not quite the same as that of UML. The UML profile extension mechanism is appropriate when a relatively small adaptation is required. When the specialisation is more extensive, a new metamodel should be defined. Hence, a new version of UML-B has now been developed. It is a UML-like formal modelling language rather than a specialisation of the UML.

UML-B[2] described in this chapter is a graphical formal modelling notation based on UML and Event-B. Event-B is a formal notation evolved from classical B, which emphasises the incorporation of an event perspective to support reactive

[2] This work is part of the EU funded research project: IST 511599 RODIN (Rigorous Open Development Environment for Complex Systems).

system development. UML-B uses the Event-B formal modelling language and its associated tools for formal verification, model-checking and animation (Abrial et al., 2006). UML-B's modelling environment includes a built-in translator U2B, which generates an Event-B model from a UML-B model. The Event-B model is analysed and verified by built-in verification tools. Verification errors are fed back and displayed on the UML-B model. This process happens automatically whenever the UML-B model is saved.

UML-B provides a top-level *Package* diagram for showing the structure and relationships between components in a project. The components include the Event-B machines and contexts. Contexts are described in a *Context* diagram, which is similar to a *Class* diagram but it has only constant data and the associated constraints. Machines are specified in a *Class* diagram. Hierarchical statemachines, which appear in *State* diagrams, can be attached to classes to describe their behaviour. A notation, μB (micro B) that borrows from the Event-B notation is used for textual constraints and actions. μB has an object-oriented style dot notation that is used to show ownership of entities such as attributes and operations by classes.

To give a flavour of UML-B, consider the specification of the telephone book in the Figure 5.1 below. The classes, NAME and NUMB represent people and telephone numbers respectively. The association role, *pbook*, represents the link from each name to its corresponding telephone number. Multiplicities on this association ensure that each name has exactly one number and each number is associated with at most one name. The *Properties* view shows μB conditions and actions for the *add* event. The *add* event of NAME class adds a new name to the class. It non-deterministically selects a *numb*, which must be an instance of the NUMB class but not already used in a link of the association *pbook*. These constraints are illustrated in the *Guards* field. It then uses this as the link for the new instance, as demonstrated in the *Actions* field. The *remove* event has no μB action, as its only action is the implicit removal of *self* from the NAME class. This specification is equivalent to the Event-B model shown in the Figure 5.2 below, which is generated by U2B automatically.

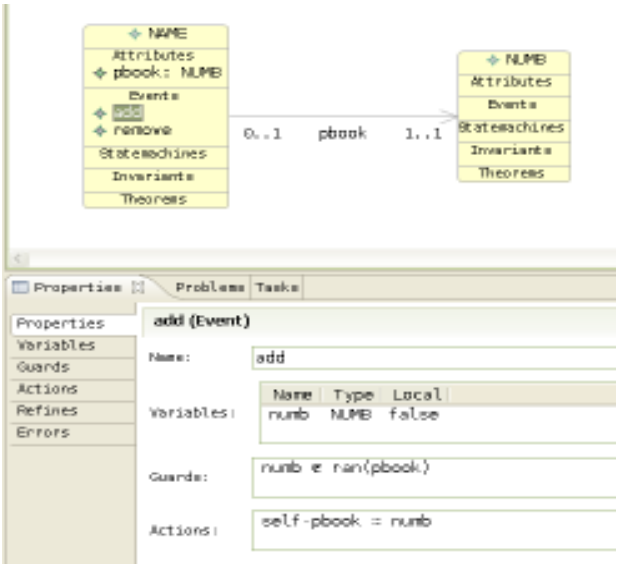


FIGURE 5.1: UML-B specification of a phone book

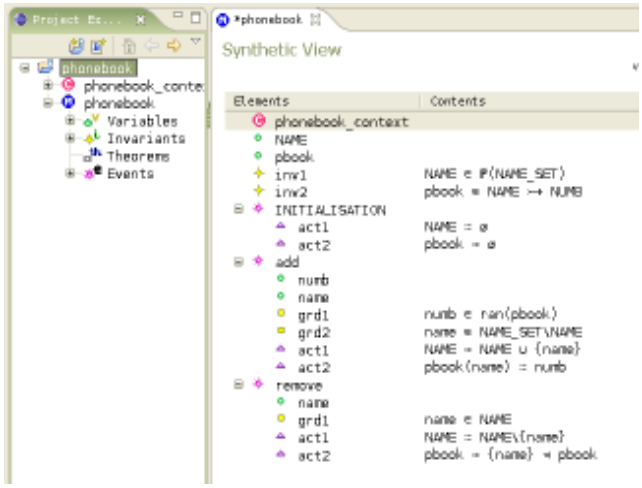


FIGURE 5.2. Event-B specification of a phone book

5.2. Theoretical Background

The theoretical background for the experiment is based on the Cognitive Theory of Multimedia Learning (Mayer, 2001). Multimedia in the theory refers to the

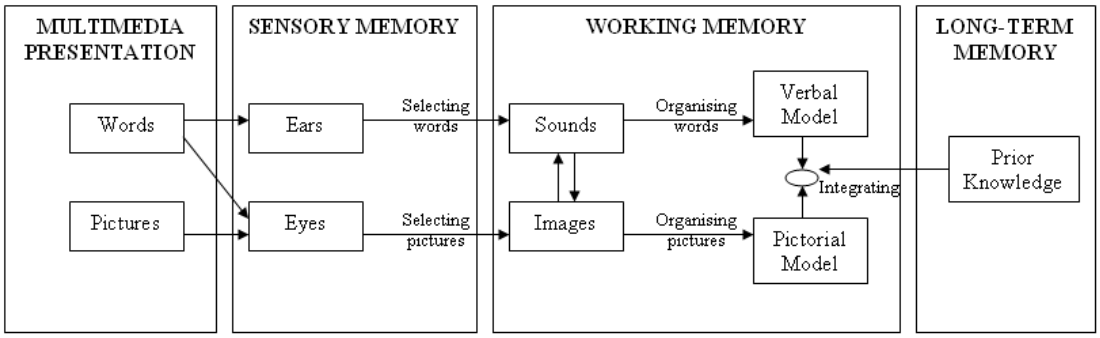
presentation of material using both words and pictures. The premise of the theory is that one can better understand an explanation when it is presented in words and pictures than in words alone. The process of multimedia learning is viewed as building coherent knowledge structures or mental models in the memory. The goal is to help one to understand the presented material and to be able to use what has been learned in other novel situations.

The theory was adopted in the experiment because it aimed to investigate the impact of the notation used in the UML-B model on the stakeholders' understanding of a problem domain. In many aspects, understanding a problem domain and the characteristics of the UML-B model itself coincide with the concepts demonstrated by the theory. Specifically, understanding a problem domain can be seen as a learning process where stakeholders gradually gain better knowledge about the domain by viewing and working on it. A UML-B model, which incorporates the use of graphical and textual representations of UML and Event-B respectively, is a multimedia material by definition.

In the theory, words (textual) and pictures (graphical) are qualitatively different by their natures. Words are useful for presenting representations that are more formal and require more effort to translate. On the other hand, pictures are more useful for presenting more intuitive and natural representations. Words and pictures complement each other but they cannot be substituted for one another. In this context, the theory seems to suggest that a UML-based formal model such as UML-B is more usable than its counterpart as its graphical representation could present its formal textual representation more intuitively and naturally. Understanding occurs when one is able to build meaningful connections between both representations and other prior knowledge in the mind.

The Cognitive Theory of Multimedia Learning integrates three other cognitive theories, which include Dual-coding Theory (Pavio, 1986; Clark et al., 1991), Cognitive Load Theory (Chandler et al, 1991; Sweller, 1999) and Working Memory Model (Baddeley, 1986; 1992; 1999). There are three primary assumptions in the theory. Firstly, words and pictures are processed through separate and distinct information processing channels. Secondly, each processing

channel is limited in its ability to process information. Thirdly, processing information in channels is an active cognitive process designed to construct coherent mental models or knowledge structures (Wittrock, 1989; Mayer, 1999). The active cognitive process involves selecting the relevant information, organising the selected information and integrating the organised information with prior knowledge in the working memory. According to the theory, the act of building connections between mental models is an important step in conceptual understanding. The Figure 5.3 below illustrates the active processing in the Cognitive Theory of Multimedia Learning.



Note: ⟶ Flow of information

FIGURE 5.3: The Cognitive Theory of Multimedia Learning (Mayer, 2001)

The construction of knowledge structures or mental models in the working memory involves three elements, namely the content of the presented material, the way in which the content is presented and the individual characteristics of the person viewing the material (Mayer, 1989a). The interaction of these three elements mainly happens in one's mind. Therefore, it is not directly indicated unless by observing the performance of the individuals involved in cognitive activities such as comprehension and problem solving tasks. As a result, such observations need to be empirical in nature (Gemino et al., 2003).

According to the Cognitive Theory of Multimedia Learning, the major goals of learning are remembering and understanding (Mayer, 2001). Remembering is defined as the ability of learners to recognise the presented information. On the other hand, understanding involves the construction of coherent mental models from the presented information. Thus, it is reflected in the ability of learners to

apply the presented information in novel situations. These abilities can be assessed through *Retention* and *Transfer* tests respectively. In a *Retention* test, learners dictate the information that is readily available in the presented material. In contrast, a *Transfer* test requires learners to provide solutions to problems which information is not explicitly presented in the material. Based on these two tests, three outcomes of learning can be determined which are *No learning*, *Rote learning* and *Meaningful learning*. *No learning* occurs when both *Retention* and *Transfer* tests are low. This implies that learners may be unsuccessful in selecting the relevant information to be processed further by the working memory. *Rote learning* occurs when the *Retention* test is high but the *Transfer* test is low. This may indicate that the relevant information has been selected and organised in the working memory but it has not been well integrated with prior knowledge. Finally, the *Meaningful learning* occurs when both *Retention* and *Transfer* tests are high. The learners' ability to perform well in both tests indicates that the relevant information has been successfully selected, organised and integrated with prior knowledge in the working memory.

The Cognitive Theory of Multimedia Learning has been developed through many years of empirical work using various experimental instruments and data (Mayer, 2001). Several studies have employed the theory to compare textual and graphical representations in science learning (Mayer, 1989b; Mayer et al., 1990; 1996), animation and narration (Mayer et al., 1991; 1992), multimedia materials (Lim et al., 2002) and conceptual models (Gemino, 2004; Gemino et al., 2005).

5.3. Research Question and Hypotheses

The main objective of this experiment was to evaluate the comprehensibility of the notation contained in a UML-B model compared to an Event-B model. A UML-B model comprises the UML-like diagrams, namely the *Package*, *Context*, *Class* and *State* diagrams, and the formal notation of Event-B. An Event-B model comprises only the Event-B notation, which is very similar to the classical B.

The experiment was conducted to confirm or refute a theory that suggests the dual representation of words (textual) and pictures (graphical) used in the UML-B model are better than the words (textual) alone used in the Event-B model. It extended Experiment 1 by investigating both models comprehensibility through their ability to foster stakeholders' understanding of the presented problem domains. Stakeholders in this context include clients and software developers who view the models for validation and maintenance (enhancement) purposes. Since the technologies are new, the stakeholders are assumed to be new users with limited hours of formal training, namely less than ten hours. Clients are assumed to have a reasonable amount of knowledge of software development and the technologies involved.

The experiment had the following broad and specific research questions:

Broad Research Question: Does a visual formal model (words and pictures) foster or promote better understanding of a problem domain than an equivalent formal model (words)?

Specific Research Question: Does a UML-B model foster or promote better understanding of a problem domain than an Event-B model?

To answer the above questions, the standard statistical inference and hypothesis testing was adopted in this experiment. The testing involves the construction of null (H_0) and alternative (H_1) hypotheses. The null hypothesis stated for this experiment was:

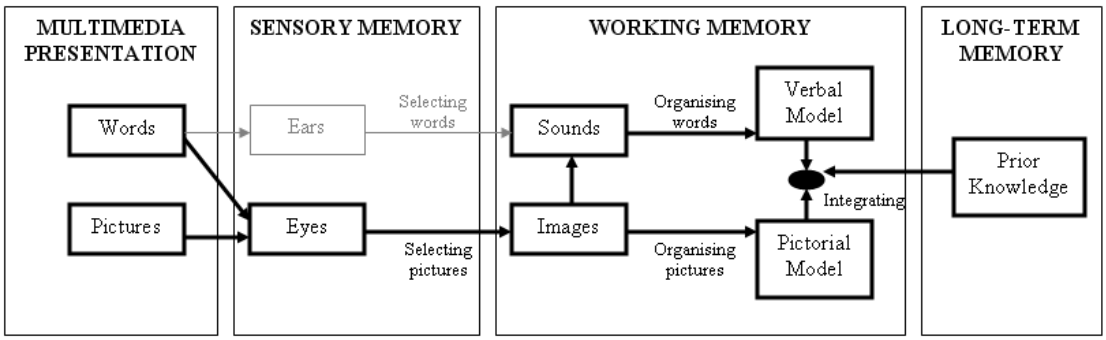
Null Hypothesis (H_0): The UML-B model is no better than the Event-B model in fostering problem domain understanding

to be rejected in favour of the alternative hypothesis:

Alternative Hypothesis (H_1): The UML-B model is better than the Event-B model in fostering problem domain understanding

A one-sided alternative hypothesis was employed in this experiment as it aimed to assess the efficacy of the UML-B model as compared to the Event-B model. The Cognitive Theory of Multimedia Learning has enabled a presumption that a UML-model should be more comprehensible than an Event-B model. The basis for this presumption is that a UML-B model guides its viewers to build a verbal mental model and a pictorial mental model of the presented information and to build connections between the two. The UML-B model allows viewers to hold corresponding verbal and pictorial representations in the working memory at the same time. This could increase the chances that viewers would be able to build mental connections between both representations and integrate them with prior mental models from the long-term memory.

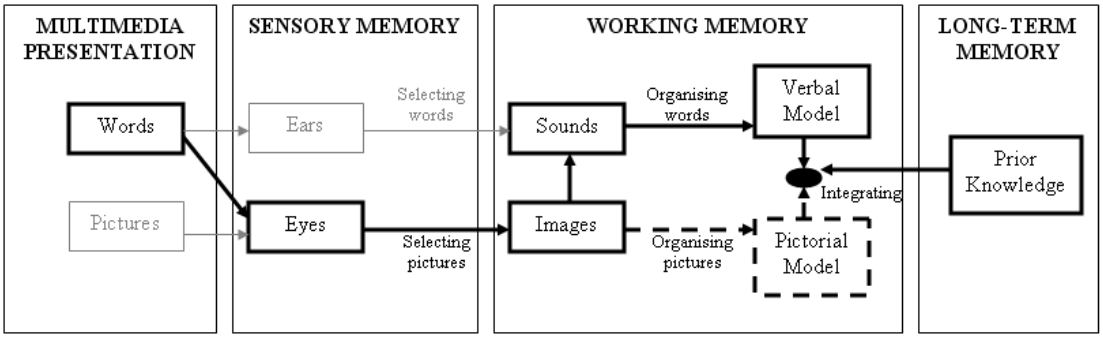
The Figure 5.4 below illustrates why a UML-B model was hypothesised to be better in terms of fostering problem domain understanding. The information presented by a UML-B model, which contains words and pictures flows into the eyes. The words and pictures then become images in the working memory. The images from pictures are organised into pictorial models, where the pictures change from the basis of images to the basis of meaning. Meanwhile, the images from the printed words are transformed as sounds in the working memory through *phonological loop* (Baddeley, 1986). The idea of *phonological loop* is that the working memory processing for verbal information involves a "mind's voice" and a "mind's ear". When visually presented verbal information such as printed word is encoded, the word is "voiced" into a sound-based or auditory-phonological code. This means the word is voiced and heard internally in the working memory, which happen continuously as long as it is needed. The sounds are then organised into verbal models where the words change from the basis of sounds to the basis of meaning. The verbal and pictorial models are then integrated with prior knowledge to form a meaningful understanding of the problem domain.



Note: ⟶ Flow of information

FIGURE 5.4: The hypothesised cognitive processing of a UML-B model

A similar process was assumed to happen in an Event-B model for the printed words, as illustrated in the Figure 5.5 below. Unlike UML-B, an Event-B model does not have pictures. Thus, most of the images resulting from the eyes are transformed as sounds and later as verbal models in the working memory. There is possibility where some word images may be transformed as pictorial models. For example, a relation between two sets is visualised mentally as a physical arrow between two bubbles containing elements. However, they are not as much as in the UML-B model, as illustrated by the black dotted lines. Therefore, the information presented in the Event-B model is heavily processed in one channel. This leads to unbalanced processing between the two channels where one is overloaded and the other is underused. As a result, the mental models are not well developed in the working memory.



Note: ⟶ Flow of information

FIGURE 5.5: The hypothesised cognitive processing of an Event-B model

5.4. Design

The experiment replicated the design employed in Experiment 1. In particular, it had a related within-subject design where each of the subjects was trained and assigned a task on both the UML-B and Event-B models. The subjects were allocated randomly into two groups, namely *Group Alpha* and *Group Beta*, by using blocking and balancing techniques. Each group had a mixture of subjects from three blocks of ability on the object-oriented technology and formal methods. The subjects were equally distributed across the two groups, thus the groups were considered as equivalent.

The design which is called cross-over trial (Senn, 2002) was employed once again to eliminate any task direction bias and ability effect. Moreover, UML-B and Event-B are new technologies where the availability of potential subjects is very limited. The design helps in obtaining a number of observations between the two treatments. In one session, *Group Alpha* was assigned a task on the UML-B model while *Group Beta* was assigned the same task on an equivalent Event-B model. The reverse was then carried out later where *Group Beta* was assigned a task on the UML-B model while *Group Alpha* was assigned the same task on an equivalent Event-B model. The Table 5.1 below shows the group and task allocation for the two sessions. There was a break between the sessions.

	Group Alpha	**Group Beta**
1st session **Case 1 (i.e. Auction System)**	Tasks on UML-B model	Equivalent tasks on Event-B model
=== BREAK ===		
2nd session **Case 2 (i.e. Library System)**	Tasks on Event-B model	Equivalent tasks on UML-B model

TABLE 5.1: Group and task allocation

5.5. Subjects

There were thirty-six students that participated in the experiment. This included eighteen third-year Undergraduate students and eighteen Masters students of Computer Science and Software Engineering courses at the University of Southampton, United Kingdom. They were students from Europe and Asia. The international students, who came from outside the United Kingdom constituted half of the subjects and the proportion of women to men was 1:4.

The subjects were students who registered for the "Critical System" course in Spring 2007 (ECS, 2007). They were taught formally on the classical B for about eight hours, one hour on Event-B and one hour on UML-B. All subjects had gone through courses on the object-oriented technology and formal methods at some points of their studies. While the subjects were familiar with UML and had been taught on B, Event-B and UML-B, they could not be considered as experts.

The experiment adhered to the University's ethical policies and guidance for conducting research involving human participants (UoS, 2007). In particular, the materials and procedure used in the experiment had been reviewed and approved by the University's Ethics Committee. A *Participant Information Sheet* and a *Consent Form* were distributed during the experiment. The tasks performed in the experiment were aligned with the expectation of the course and had pedagogical values. The level of understanding tested in the experiment was useful for the coursework and examination. It acted as revision on what had been taught in the course and a space for reflection on the learning.

The subjects were divided randomly and equally into two groups by using blocking and counter-balancing techniques. There were eighteen students in both *Group Alpha* and *Group Beta*. Each group consisted of nine Undergraduate students and nine Masters student. All the subjects were aware that the experiment was intended for research purposes and had agreed to participate.

The subjects were in the final semester of their respective courses and had reasonable amount of experience and knowledge of software development. Some of the Masters students had some work experience for at least one year. They were the next generation of professionals. Thus, they represented closely the population under study; software stakeholders.

5.6. Variables

The independent variable of the experiment was the notations used in the UML-B and Event-B models. Similar to Experiment 1, the experiment aimed to examine the effect of the notations on the efficiency in performing the comprehension tasks. Efficiency by itself cannot be measured directly. It can only be measured indirectly in terms of the accuracy and duration of the tasks. Therefore, two direct measures that reflected those aspects were used. As these measures were expected to change by applying the treatments, they were considered as the dependent variables of the experiment.

The identified dependent variables were *Score* and *Time Taken*. The units for the *Score* and *Time Taken* were *marks* and *minutes* respectively. These two variables were used to measure the *Rate of scoring*, which unit was *marks/minute*. The *Rate of Scoring* was indeed the measure of comprehension efficiency. The selection of *Score* and *Time Taken* as dependent variables and the *Rate of Scoring* as a meaningful measure of comprehension efficiency have been explained in detail in **Chapter 3**.

The following sub-sections provide specific explanation of the measures used in the experiment.

5.6.1. Direct Measures

The experiment identified two direct measures that acted as its dependent variables as follows:

- **Score**

This variable is the mark obtained. There were six questions altogether. Three questions were given specific allocation of marks as the answers were mainly based on what were illustrated in the models. The other three questions involved the application of model understanding to propose novel solutions. The marking was based on as many reasonable and logical answers given by the subjects. Some basic answers however were expected from the subjects on these questions. Thus, a list of acceptable answers was also prepared. In addition, extra marks were also given to creative answers.

The questions had been carefully constructed so that the marks could be easily decided. An answer sheet of acceptable answers was prepared beforehand. One person did the marking so that there was consistency throughout the process. The marking process was repeated twice to ensure its accuracy. As the marker was an independent user of both UML-B and Event-B, no bias was introduced on any model.

Each question was marked individually. Besides giving the *Score* for each question, this allowed them to be grouped according to specific measures. For examples, the marks for all questions were combined together to determine the overall *Score*, the marks for *Question 1*, *Question 2* and *Question 4* determined the *Score* for the *Retention* test and the marks for *Question 3*, *Question 5* and *Question 6* determined the *Score* for the *Transfer* test. The detailed elaboration on these tests is included later in this section.

- **Time Taken**

This variable is the time taken to answer each question in minutes. Subjects were required to state the *Start time* and *End time* for each question. Subtracting the *Start time* from the *End time* produced the *Time Taken*. Similar to the *Score*, the *Time Taken* was measured individually so that it could be combined as necessary to suit the measures of interest.

5.6.2. Indirect Measures

The experiment investigated two main aspects of comprehension efficiency, namely remembering and understanding. One involved the recognition of information and the other was the construction of meaningful understanding from the presented information to provide novel solutions. The former was measured based on the subjects' ability to elicit the relevant information from the models. The latter was measured based on the subjects' ability to solve problems. In the Cognitive Theory of Multimedia Learning, the mechanisms to determine those abilities are called the *Retention* and *Transfer* tests respectively.

As described in the earlier section, the *Retention* and *Transfer* tests can be used to determine three learning outcomes as illustrated in the Table 5.2 below. Unlike previous studies that used the Cognitive Theory of Multimedia Learning (Mayer et al., 1990; 1991; Lim et al., 2002; Gemino, 2004; Gemino et al., 2005) or its measures (Bodart et al., 2001; Burton-Jones et al., 2002), this experiment considered the *Score* and *Time Taken* rather than *Score* alone in both the *Retention* and *Transfer* tests. This was because the *Score* and *Time Taken*, which constituted the *Rate of Scoring*, measured the efficiency of the comprehension tasks.

The experiment aimed to compare the relative comprehensibility of two models. Therefore, the learning outcome was determined based on whether the differences in the *Rate of scoring* between the two models were significant. As the experiment employed a one-sided alternative hypothesis testing in favour of the UML-B model, any significant differences detected would indicate that the UML-B model

is better than the Event-B model. This means if the differences were significant in both the *Retention* and *Transfer* tests, the UML-B model could then be regarded as being better than the Event-B model in promoting *Meaningful learning*. If the differences were significant in the *Retention* test but not in the *Transfer* test, the UML-B model could be regarded as being better than the Event-B in promoting *Rote learning*. In this sense, the *Rote learning* would mean that the UML-B model is better than the Event-B model in allowing subjects to recognise the relevant information. However, it could not suggest that the UML-B model is better than the Event-B model in terms of subjects' ability to provide novel solutions. The opposite effect where the differences were significant in the *Transfer* test but not in the *Retention* test was not expected. This was because the *Transfer* test was dependent on the subjects' ability to integrate what were in the models with other elements beyond the models. Therefore, it was impossible to be good in the *Transfer* test but not in the *Retention* test. The insignificant differences in both tests would suggest that the UML-B model is no better than the Event-B model in promoting learning. *No learning* in this context should not be interpreted as no knowledge at all but rather there is no better effect that one model could bring than the other in terms of understanding.

	Performance			
Learning Outcome	**Retention Test**		**Transfer Test**	
No Learning	Low	Not significant	Low	Not significant
Rote Learning	High	Significant	Low	Not significant
Meaningful Learning	High	Significant	High	Significant

TABLE 5.2: Relationship between performance of Retention and Transfer tests and learning outcomes

5.6.3. Other Measures

Besides quantitative measures above, the experiment also employed some qualitative measures. The measures included the comprehension and problem solving strategies used by the subjects to answer the questions, the subjective rating of model comprehensibility, the subjects' preference between the model notations and the subjects' personal comments on the models. All questions except the problem solving strategies were also used in Experiment 1. The

detailed elaboration on the instruments used to measure these aspects is included in the next section.

5.7. Materials and Procedure

5.7.1. Design of the Materials

The materials used in the experiment included models written in each notation and a questionnaire on each of the models. There were also a *Participant Information Sheet* and a *Consent Form*, an instruction sheet that explained the steps required when performing the tasks, a set of answer sheets to write down the answers and a list that contained B symbols and their description.

Four models that represented two separate case studies were developed to avoid learning effects. The same case studies used in Experiment 1 were employed in this experiment, namely *Auction System* and *Library System*. In the first session, *Group Alpha* was given a UML-B model and *Group Beta* was given an equivalent Event-B model on the first case study. In the second session, *Group Alpha* was given an Event-B model and *Group Beta* was given an equivalent UML-B model on the second case study. Both the UML-B and Event-B models for each session were made *informationally equivalent* (Larkin et al., 1987). This means the information contained in one model was also inferable from the other. It was the models' *computational equivalent* (Larkin et al., 1987) that this experiment aimed to measure. Two *informationally equivalent* models are *computationally equivalent* only if any inference that can be drawn easily and quickly from the information in one model can also be drawn easily and quickly from the information in the other model. Even though both the UML-B and Event-B models were *informationally equivalent*, one would expect differences in efficiency of performing the comprehension tasks if they were not *computationally equivalent*.

The models for the second session were made as equivalent as possible to the first session so that the treatment effect to be tested remained the same. On the other hand, the models were made different enough in subject matter to avoid confounding the second session with learning gained from the first session. In each case study, the UML-B model had one *Package* diagram that contained one *Context* diagram, one *Class* diagram with two classes and two *State* diagrams with two states each. There were about 130 lines of script for each of the Event-B models. The models can be found in the **Appendix C**.

5.7.1.1 Questions on Models

Understanding involves the formation of coherent mental models in the working memory. Mental models are general knowledge structures that encapsulate numerous elements of information into a single meaningful element (Smith et al., 2007). In the context of material that contains words and pictures, the mental models are formed when the verbal and pictorial models developed in the working memory can be successfully integrated with prior knowledge to form a meaningful understanding (Mayer, 2001). Only after this formation, the mental models can be organised and held in the long-term memory into a manner in which it can be widely used.

In order to empirically evaluate the formation of mental models in the working memory, it is necessary to explore some of the typical ways that knowledge can be structured. There are five common types of knowledge structures used in scientific text (Cook et al., 1988; Chambliss et al., 1998), as summarised in the Table 5.3 below. The formation of knowledge structures can be demonstrated by the ability of the subjects to explain cause-and-effect chains, compare and contrast two elements along several dimensions, describe main ideas with supporting details, list a set of collection of items and analyse a domain into sets and subsets. Understanding often requires constructing one of these kinds of knowledge structures.

Structure	Description	Task required
Process	Connecting series of events	*Explain* a cause-and-effect chain
Comparison	Examining the relationship between two or more things	*Compare and contrast* two or more elements
Generalisation	Extension of main ideas	*Describe* main ideas and supporting details
Enumeration	Listing of facts	*List or present* a number of items
Classification	Grouping items	*Analyse* a category into sub-categories

TABLE 5.3: Five types of knowledge structures in scientific text

Stakeholders interpret and conceptualise the information presented in the specifications to tackle problems that are often too big, too ill defined and too complex for easy solution (DeGrace et al., 1998). Besides knowledge construction, understanding a problem domain can thus be viewed as a process of cognitive learning. According to the Bloom's Taxonomy (Bloom et al., 1956), cognitive learning is demonstrated by the knowledge recall and the development of intellectual skills. These include comprehending information, organising ideas, analysing and synthesising data, applying knowledge, choosing some alternatives in problem solving and evaluating ideas or actions. The taxonomy identified six levels, from the simple recall or recognition of facts through increasingly more complex and abstract mental levels, as shown in the Table 5.4 below. Research has confirmed that the first four levels are hierarchical (Hummel et al., 1994). The ordering of the two highest levels, namely *Synthesis* and *Evaluation* however, has been proposed to be reversed (Anderson et al., 2000) and also to be at the same level (Huitt, 2004). Nevertheless, research has confirmed that both are necessary for successful problem solving (Huitt, 1992). When either is omitted during the problem solving process, effectiveness declines.

	Cognitive Level	Description
Less complex ‖ ‖ ‖ ‖ ‖ ‖ ‖ \‖/ **More complex**	**Knowledge:** *Information Gathering*	Remembering and recalling of appropriate, previously learned information
	Comprehension: *Confirming*	Understanding the meaning of informational material
	Application: *Making Use of Knowledge*	Using previously learned material in new and concrete situations
	Analysis: *Taking Apart*	Breaking down informational materials into their component parts
	Synthesis/Creating: *Creatively Putting Parts Together*	Creatively or divergently applying prior knowledge and skills to produce a new or original whole
	Evaluation: *Judging the Outcome*	Judging the value of material

TABLE 5.4: Six cognitive levels of the Bloom's Taxonomy

The experiment combined the ideas of knowledge structures and the Bloom's Taxonomy as its measurement instrument. In many ways, both ideas fit together and focus on similar aspects of intellectual skills. Because of this reason, the questions used in the experiment were constructed based on these two ideas. The Table 5.5 below illustrates the mapping between the questions of the first case study, *Case 1*, and these ideas as an example. There were six questions altogether for each model. In each case study, the questionnaires on both the UML-B and Event-B models were similar. Moreover, the same types of questions were asked in both two case studies. The questions were open-ended in nature. This was to ensure the subjects could derive the answers freely based on genuine understanding of the presented models.

Question 1 was intended to assess whether the subjects could be able to capture the main ideas of the presented problem domain or system. This question also served to ensure that the subjects scanned the whole model so that they would be ready for answering more specific questions later. *Question 2* provided the subjects with a possible scenario that might happen in real life when such a system is used. The model indeed contained the necessary elements that could handle such scenario. The question therefore assessed whether or not the subjects could recognise the elements and understand how they work. *Question 3* required the subjects to modify and enhance the model so that it could incorporate a new feature. *Question 4* indicated that the model contained two operations that were similar in functionality but did not actually react under the same conditions and

actions. Thus, the subjects were required to compare the operations by dictating the differences. *Question 5* provided the subjects with a scenario that was not catered by the model. The subjects had to formulate a plan to suit the scenario while considering the existing elements illustrated in the model. *Question 6* required the subjects to propose ways to improve the presented problem domain so that it could represent a better system. The subjects were instructed to provide as many as possible solutions that they could think of to the question. The subjects were not restricted to answer the six questions in any particular order. However, the questions had been organised in an increasingly complex order based on the Bloom's Taxonomy (Bloom et al., 1956), Therefore, it was more logical if they subjects answered the questions in the given order.

As explained earlier, the mechanism used in the experiment to measure the efficiency of comprehension tasks was through the *Retention* and *Transfer* tests. Three questions were constructed to assess the ability of the subjects to recognise some aspects of the problem domain presented by the model. The questions, namely *Question 1*, *Question 2* and *Question 4* acted as the *Retention* test. The other three questions, namely *Question 3*, *Question 5* and *Question 6* required the subjects to propose solutions that were not indicated in the model. These questions formed the *Transfer* test. The *Transfer test* was indeed the critical assessment, which determined whether the subjects could be able to integrate the organised mental models in the working memory with other knowledge from the long-term memory.

Each question required the subjects to provide some kind of elaboration. This ensured that deep understanding was tested rather than just brief understanding. In addition, most questions required subjects to view several parts of the models so there was no bias placed on any specific characteristics of the notations used in the models. The questions were designed to be challenging in a reasonable manner. Generally, each question was estimated to take between 5 to 6 minutes to complete.

Questions	Knowledge Structure	Cognitive Level
Question 1: Describe briefly in natural language the main ideas illustrated in the given model (i.e. a high level description/summary of the key entities involved and the relationships/interactions between them).	Enumeration Generalisation	Knowledge Comprehension
Question 2: Consider the following situation: <*Scenario*> How does the given model handle this sort of situation? Explain.	Process	Comprehension
Question 3: A new requirement is added to the system: <*Requirement*> How would you enhance the given model to include this new requirement?	Process Comparison	Application
Question 4: There are two different situations/restrictions of … in the given model. Compare and contrast them by clearly explaining the differences.	Comparison Classification	Analysis
Question 5: <*Scenario*> What are the conditions for this scenario to happen? What actions should be taken by the system? Propose an idea (plan) in natural language for meeting this requirement.	Process Comparison Generalisation Enumeration Classification	Synthesis
Question 6: Criticise the given model. Suggest ways (as much as you could think) that the model should be improved to represent a better system.	Process Comparison Generalisation Enumeration Classification	Evaluation

TABLE 5.5: The mapping of questions with knowledge structures and the Bloom's Taxonomy

5.7.1.2. Debriefing Questions

Question 5 and *Question 6* of both models in both sessions contained debriefing questions. These two questions required the subjects to think beyond the models, which the answers were not directly indicated in the models. For example, *Question 5* required subjects to propose a plan on how the system should behave in a specific scenario whereas *Question 6* asked them to criticise the current models. The objective of these debriefing questions was to identify the strategies used by the subjects when they were confronted by a novel situation and must

decide on course of action. Strategies that are normally used by humans in solving problems can be classified into four main types. They are applying weak methods to search heuristically for a possible solution, using analogical transformation to a known solution of a similar problem, instantiating specific plans and applying general plans to reduce the problem (Ryszard et al., 1986; Smith et al., 2007). The nature of the debriefing questions was multiple selections, as solving problems could involve using a combination of the above approaches (Ryszard et al., 1986). The subjects were asked to select one or more of these options. They were allowed to state other answers if the answers did not belong to any of the given options.

Besides the debriefing questions on the problem solving strategies, there were four other types of debriefing questions that asked subjects' opinions or perception of the UML-B and Event-B models. One debriefing question investigated the direction of comprehension employed by the subjects when understanding both UML-B and Event-B models. Specifically, the question asked whether the subjects employed the *Top-down* (Brooks, 1983, Soloway et al., 1982) or *Bottom-up* (Schneiderman et al., 1979, Pennington, 1987) strategy when understanding each model. This question was intended to identify whether the notation had any particular influence on the way subjects understood the models.

Secondly, there were also debriefing questions on model comprehensibility. The subjects were asked to rate each model's comprehensibility by using a symmetric ordinal scale or Likert scale. The questions focused on the models' "ease of understanding" rather than "ease of use". Besides the selection on the Likert scale, the subjects were also required to state which parts of the models that they thought would ease and hinder the understanding. The aim of these questions was to identify the impacts of the models, particularly the characteristics of the notations used, on the process of understanding.

The third debriefing question was about the subjects' preference of model notation. The subjects were required to state which model between UML-B, Event-B or classical B that they were more comfortable to work with. They had to select one option and justify their preference. This question aimed to identify

which model was more attractive to the subjects and what aspects that made it more attractive than the others. This would also give some preliminary indication of the methods' adoption in future and ideas for further improvement.

The last debriefing question gathered the subjects' personal comments on any aspects of the models and the experiment. This included both positive and negative comments. The question was intended to identify any issues and to collect suggestions for improvement.

The debriefing questions on the direction of comprehension were given at the end of the task in each session. In contrast, other debriefing questions were given to the subjects at the end of the second session when they had dealt with both models. The details of the questions used in the experiment can be found in the **Appendix C**.

As mentioned in the **Theoretical Background** section, three elements involved in understanding a material, namely the content, the presentation method and the individual characteristics of the person viewing the material (Mayer, 1989a). The experiment aimed to investigate the effect of the presentation method on the understanding. Therefore, the other elements were controlled accordingly so that they would not confound the results. The Figure 5.6 below illustrates the interaction between these elements and the experiment's context.

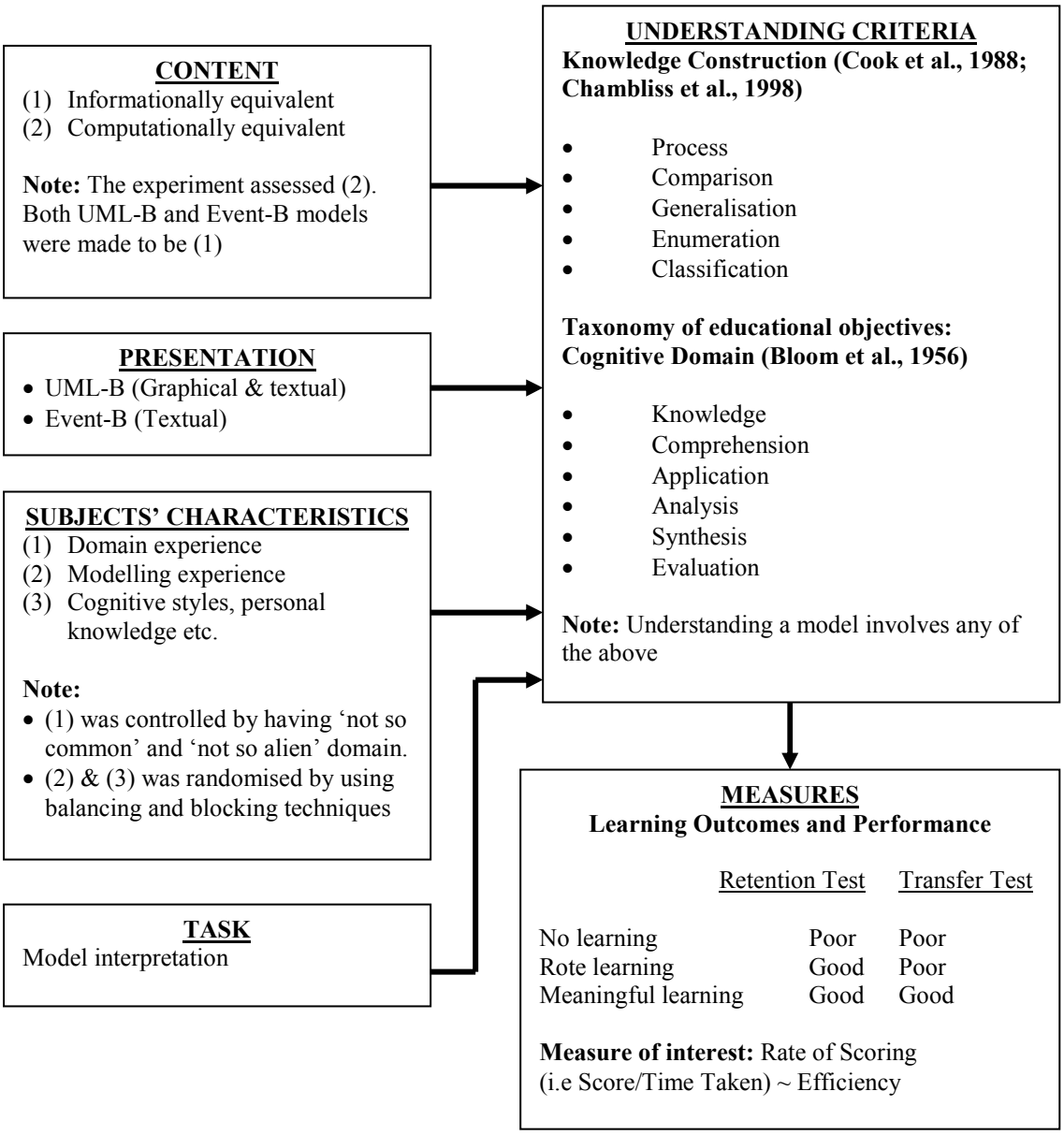


FIGURE 5.6: Overview of the experiment's context

5.7.2. Pilot study

A pilot study had been conducted to validate and verify the accuracy of the materials prepared for the experiment. These included the clarity of the instruction, the validity and complexity of the questions and the practicality of the tasks required relative to the allocated time. Besides, the pilot study was also intended to identify any unforeseen circumstances and operational issues that might have not been realised during the preparation of the materials.

Seven postgraduate students were involved in the study. Most of them were active users of the classical B and had learned the UML at some points of their studies. Three of the postgraduates had some exposure to UML-B whereas the rest knew the basic concepts of UML-B but no practical experience of using it. Similarly, three of them had some brief experience of Event-B. They were selected as participants because they reflected the characteristics of the actual subjects while possessed the necessary expertise to validate the accuracy of the materials. Some issues had been discovered during the study. These included the complexity of the tasks and the clarity of the questions and instruction. Minor changes were made to the materials and procedures based on the comments obtained from the study.

To ensure the models were accurate technically, an academic expert and two independent users reviewed the final models. Moreover, the models were also run through the verification tools. Some changes were made to the models based on the feedback.

5.7.3. Procedure

The experiment was an online exercise where the subjects viewed models on the computer screen and answered the questionnaires on the given sheets. Each subject was given a computer and an envelope in each session. The envelope for the first session contained a two-page sheet of instruction, a *Participant Information Sheet*, a *Consent Form*, a set of questionnaires and answer sheets, and a summary of B symbols. In the second session, the envelope contained only a set of questionnaires and answer sheets.

The computers used in the experiment were installed with the necessary software for running the UML-B and Event models. The operability of the computer and software systems had been tested to ensure there would be no technical faults during execution. The subjects had been briefed on how to navigate through the models using the computer during the lecture. In fact, they were also provided with a step-by-step instruction. There was a gap of one week between the lecture and the experiment. This gave the subjects some time to familiarise themselves with the environment. This was to ensure that the understanding of the models would not be confounded by the subjects' ignorance of using the software and navigating through models.

The experiment was conducted in a two-hour slot. The slot was divided into two sessions with forty-five minutes each. After the second session had ended, the subjects were given another five minutes to complete the debriefing questions that wrapped up the experiment. The remaining minutes were allocated for the subjects to settle down, read the instructions and the break between tasks, as illustrated in the Figure 5.7 below. The experiment was executed in two separate rooms. One group was located in one room and the other group in another room. There were two invigilators monitored the sessions in each room.

In the beginning of the first session, the subjects were instructed to read the *Participant Information Sheet* before signing the *Consent Form*. This was to ensure that they understood why and how the experiment would be executed and its implication. Only after they had read, understood and signed the *Consent Form*, they were allowed to participate. The subjects were not allowed to talk to each other and leave the room when the two sessions were run. However, they could do so during the break between sessions. This was to ensure that the ethics policy was adhered while minimising any threats to the validity of the results.

The environment was like an online examination where the subjects were separated from each other. They were allowed to refer to textbooks or notes. The subjects had the opportunity to inform the invigilators at all time if they had any problems on the materials and procedure. The invigilators would only assist subjects on the "external" aspect of the materials such as the meaning of a

particular question or how to get to certain parts of the model. They could not give any advice on the "internal" aspect such as the meaning of the symbols or specific information illustrated in the models. This was to avoid any bias to the results.

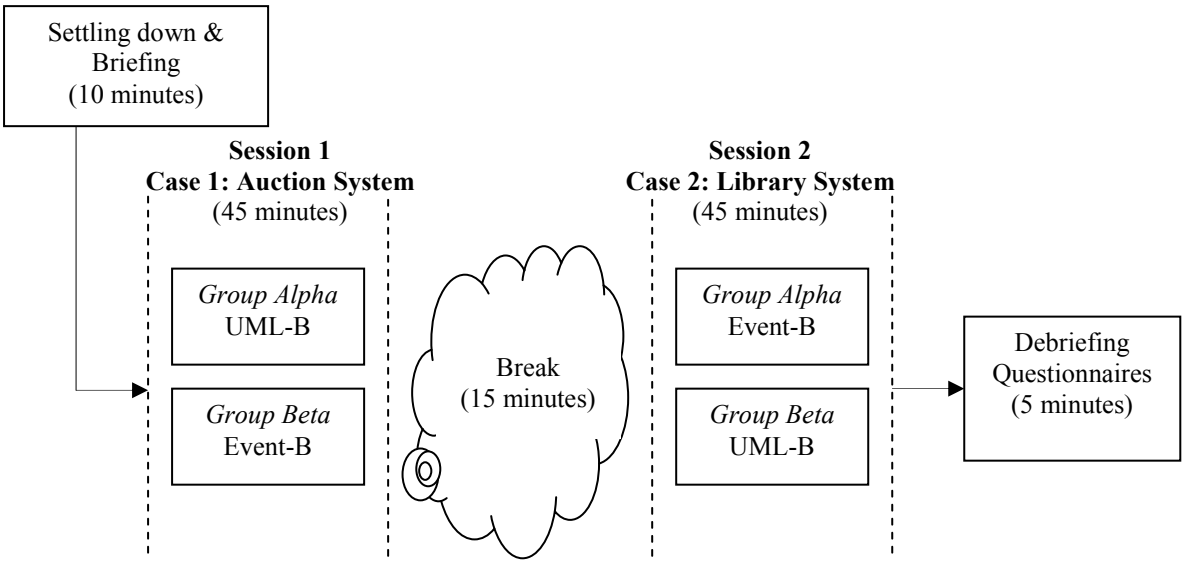


FIGURE 5.7: Overview of the experiment's procedure

There are two main approaches adopted by studies that assess understanding. One approach involves allowing subjects to understand a material in a given time and later the material is taken away from the subjects (Mayer, 2001; Gemino, 1998; Gemino et al., 2005; Bodart et al., 2001). The subjects have no access to the material and thus answer the questions based on what they can remember. The other approach allows the subjects to refer to the material throughout the sessions (Kim et al., 2000; Burton-Jones et al., 2002). This experiment adopted the second approach where the subjects had the access to the material when answering the questions. This was because in reality, most documentation would be available to stakeholders throughout the process. It would be unnecessary for stakeholders to memorise any aspects of a system unless for educational purposes. Moreover, the tasks in the experiment required the subjects to find information from various parts of the models. In fact, it involved mathematical notation. Taking away the models would complicate the tasks superfluously. It would be unlikely the subjects could remember enough within a short period of time.

5.8. Results and Analysis

The experiment employed two types of measures, namely the quantitative and qualitative measures. The quantitative measures involve the comparison of numeric data between models while the qualitative measures mainly involve the subjects' perspectives on the given tasks and models. The measures are treated and elaborated separately as follows.

5.8.1. Quantitative Measures and Analysis

As described in the previous section, the dependent variables of this experiment were *Score* and *Time Taken*. These direct measures were taken to determine an indirect derived measure namely the *Rate of Scoring*, which was obtained by dividing the *Score* by the *Time Taken*. The *Rate of Scoring* was the measure of interest in this experiment as it took into account the accuracy and the duration of the comprehension tasks. The scale used for the *Rate of Scoring* was *marks per minute* (marks/min). This means a model with a higher *Rate of Scoring* is better than otherwise as it is able to promote more accurate understanding of the presented information with least time taken.

The *Score* and *Time Taken* for each question were measured individually. This allowed the individual *Rate of Scoring* distribution for *Question 1* to *Question 6* to be determined and observed. The raw data of these measures can be found in the **Appendix C**. Subsequently, the differences in the *Rate of Scoring* between the two treatment distributions, UML-B and Event-B, with respect to each of the questions were tested. This was to determine whether or not the observed differences were significant. The calculation of confidence intervals at 95% was then pursued for the significant distributions.

The experiment had two main types of measurement and analysis, which were based on the *Retention* and *Transfer* tests. As mentioned earlier, the *Retention* test encompassed *Question 1*, *Question 2* and *Question 4*. These questions assessed the ability of the subjects to recognise some aspects of the problem domain

presented by the models. On the other hand, the *Transfer* test included *Question 3*, *Question 5* and *Question 6*. These questions required the subjects to propose solutions that were not indicated in the models. The measurement for the *Retention* test was obtained by consolidating the total *Score* and the total *Time Taken* for *Question 1*, *Question 2* and *Question 4*. Similarly, the measurement for the *Transfer* test was obtained by consolidating the total *Score* and the total *Time Taken* for *Question 3*, *Question 5* and *Question 6*. The respective statistical inference and the hypothesis analysis were then applied on these measurements.

The following paragraphs explain the descriptive statistics and analysis made on the data. Later, the statistical inference and hypothesis analysis are discussed.

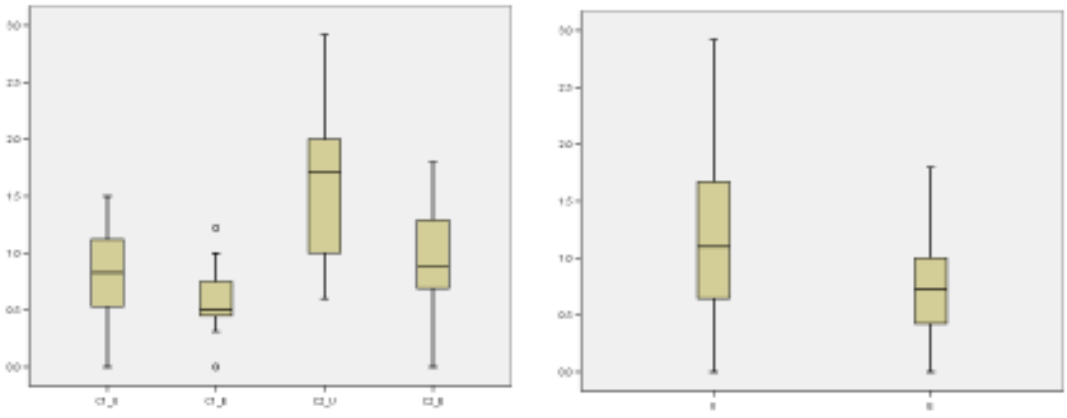
5.8.1.1. Descriptive Statistics and Analysis

The simplest useful numerical description of a distribution consists of both a measure of centre and a measure of spread (Moore et al., 2006). These measures depict how the data are distributed across the population or sample. The Figure 5.8 to 5.13 below illustrate these measures for each of the questions respectively. For all the tables, column *Min* shows the minimum values, column *1st Quarter* shows the first quartile values, column *Mean* shows the average values, column *Median* shows the middle values, column *3rd Quarter* shows the third quartile values, column *Max* shows the maximum values, column *Std Dev* shows the degree of variation, and column *N* gives the number of collected data. Rows *Case 1:UML-B* and *Case 1:Event-B* present the *Rate of Scoring* of the respective models for the *Auction System*. Rows *Case 2:UML-B* and *Case 2:Event-B* present the *Rate of Scoring* of the respective models for the *Library System*. The last two rows present the grouped *Rate of Scoring* based on the models used, regardless of the system. In addition to the tables, the respective boxplots are also included to illustrate the distributions graphically for easy viewing and comparison.

The numbers of subjects that participated in the experiment were eighteen for *Case 1:UML-B* and *Case 2:Event-B* and eighteen for *Case 1:Event-B* and *Case 2:UML-B*. This resulted in thirty-six data had been collected altogether for each model. The data considered for the analysis in each question however were

slightly less than the total collected data. This was due to data cleaning, which was conducted to ensure the validity of the analysis. In particular, the analysis excluded the subjects who did not attempt the question at all, which was identified based on the zero values (0) in both *Score* and *Time Taken*. The numbers are stated in the brackets under the *N* column in the respective tables. However, the subjects who had attempted the question for some time (non-zero *Time Taken*) but failed to get any score (zero *Score*) were included in the analysis. These data essentially resulted in zero values for the *Rate of Scoring* (0.00). These data imply that the subjects had struggled to understand the presented information or perhaps had misunderstood it. These possibilities indicate that there were some aspects of the models that hindered the subjects to capture the correct understanding of the presented information. Therefore, they were included in the analysis. The numbers of these data are illustrated in the brackets under the *Min* column in the tables.

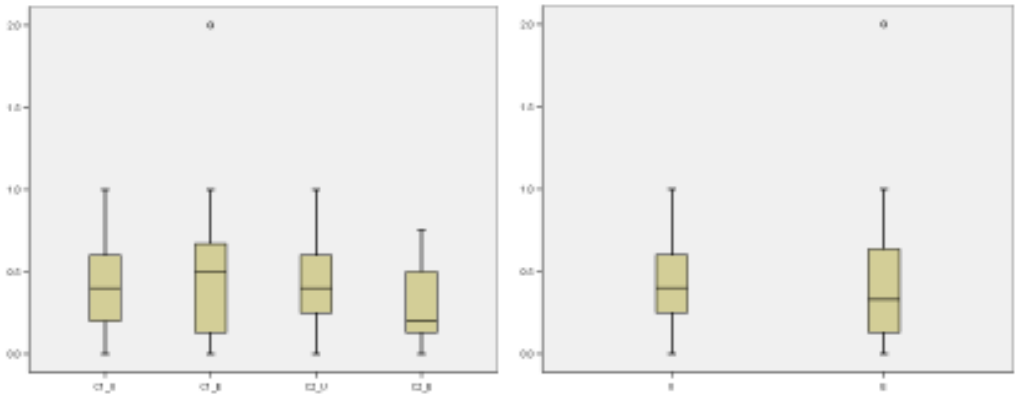
	Min	1st Quarter	Mean	Median	3rd Quarter	Max	Std Dev	N
Case 1: UML-B	0.00 (1)	0.51	0.78	0.77	1.12	1.50	0.41	18
Case 1: Event-B	0.00 (1)	0.45	0.59	0.50	0.75	1.22	0.29	17 (1)
Case 2: UML-B	0.60	1.00	1.56	1.71	2.00	2.92	0.63	17 (1)
Case 2: Event-B	0.00 (1)	0.47	0.87	0.86	1.27	1.80	0.49	18
UML-B	0.00 (1)	0.65	1.16	1.10	1.67	2.92	0.65	35
Event-B	0.00 (2)	0.43	0.73	0.73	1.00	1.80	0.42	35



Note: **C1_U** => Case 1:UML-B; **C1_E** => Case 1:Event-B; **C2_U** => Case 2:UML-B; **C2_E** => Case 2: Event-B; **U** => UML-B (Case 1 & 2); **E** => Event-B (Case 1 & 2)

FIGURE 5.8: The Rate of Scoring distribution for Question 1 (Unit: marks/min)

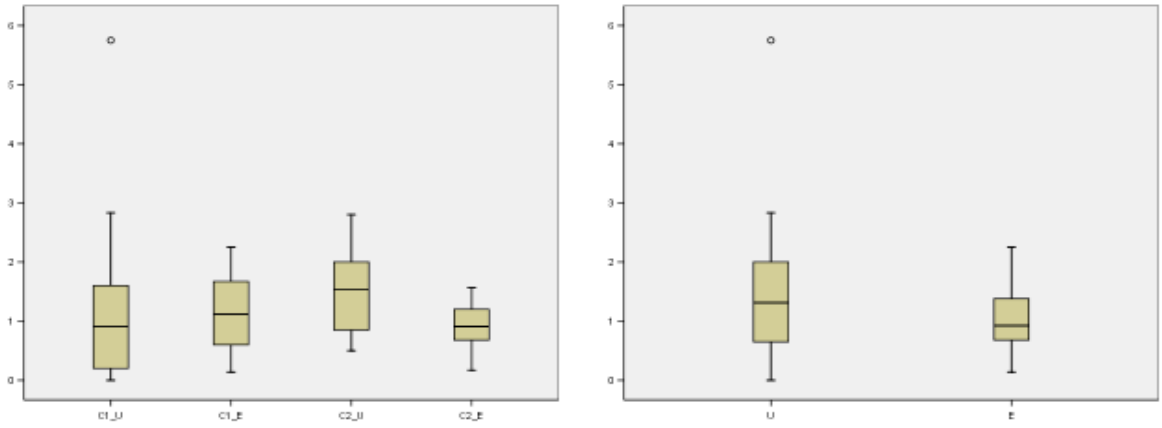
	Min	1st Quarter	Mean	Median	3rd Quarter	Max	Std Dev	N
Case 1: UML-B	0.00 (2)	0.23	0.42	0.42	0.58	1.00	0.27	18
Case 1: Event-B	0.0 (3)	0.13	0.51	0.50	0.67	2.00	0.48	17 (1)
Case 2: UML-B	0.00 (1)	0.25	0.47	0.40	0.60	1.00	0.30	17 (1)
Case 2: Event-B	0.00 (3)	0.14	0.30	0.25	0.48	0.75	0.25	18
UML-B	0.00 (3)	0.25	0.44	0.40	0.60	1.00	0.28	35
Event-B	0.00 (6)	0.13	0.40	0.33	0.64	2.00	0.39	35



Note: **C1_U** => Case 1:UML-B; **C1_E** => Case 1:Event-B; **C2_U** => Case 2:UML-B; **C2_E** => Case 2: Event-B; **U** => UML-B (Case 1 & 2); **E** => Event-B (Case 1 & 2)

FIGURE 5.9: The Rate of Scoring distribution for Question 2 (Unit: marks/min)

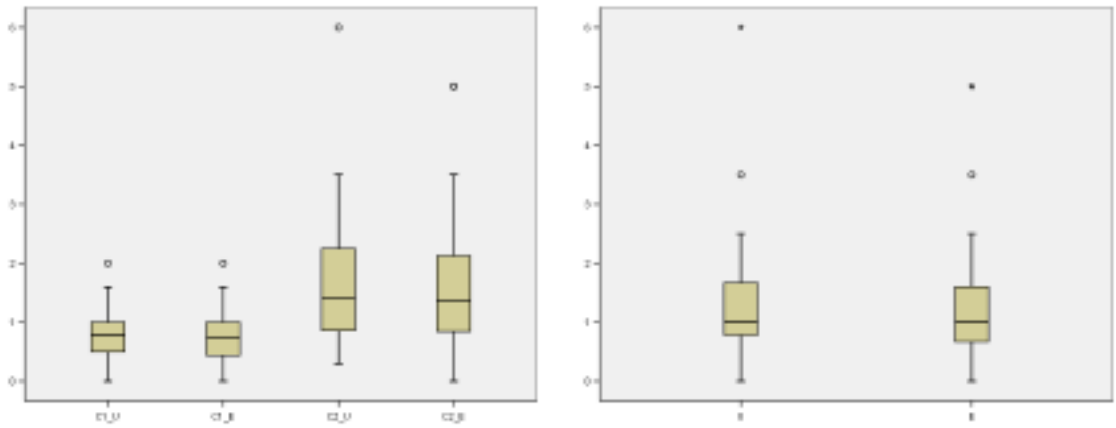
	Min	1st Quarter	Mean	Median	3rd Quarter	Max	Std Dev	N
Case 1: UML-B	0.00 (2)	0.25	1.29	0.91	1.58	5.75	1.40	18
Case 1: Event-B	0.14	0.62	1.13	1.12	1.63	2.25	0.60	14 (4)
Case 2: UML-B	0.50	0.83	1.47	1.50	2.00	2.80	0.68	15 (3)
Case 2: Event-B	0.17	0.72	0.88	0.88	1.13	1.57	0.40	16 (2)
UML-B	0.00 (2)	0.67	1.37	1.30	2.00	5.75	1.12	33
Event-B	0.14	0.68	0.99	0.93	1.36	2.25	0.51	30



Note: **C1_U** => Case 1:UML-B; **C1_E** => Case 1:Event-B; **C2_U** => Case 2:UML-B; **C2_E** => Case 2: Event-B; **U** => UML-B (Case 1 & 2); **E** => Event-B (Case 1 & 2)

FIGURE 5.10: The Rate of Scoring distribution for Question 3 (Unit: marks/min)

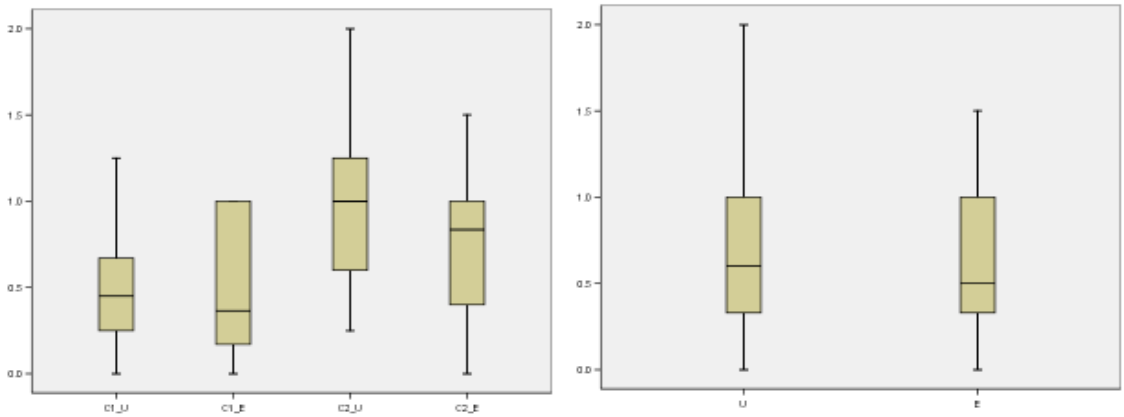
	Min	1st Quarter	Mean	Median	3rd Quarter	Max	Std Dev	N
Case 1: UML-B	0.00 (1)	0.50	0.82	0.79	1.00	2.00	0.51	16 (2)
Case 1: Event-B	0.00 (3)	0.47	0.79	0.80	1.00	2.00	0.55	17 (1)
Case 2: UML-B	0.30	0.86	1.80	1.50	2.00	6.00	1.35	17 (1)
Case 2: Event-B	0.00 (1)	0.86	1.77	1.25	1.75	5.00	1.45	17 (1)
UML-B	0.00 (1)	0.78	1.32	1.00	1.67	6.00	1.13	33
Event-B	0.00 (4)	0.67	1.28	1.00	1.58	5.00	1.19	34



Note: **C1_U** => Case 1:UML-B; **C1_E** => Case 1:Event-B; **C2_U** => Case 2:UML-B; **C2_E** => Case 2: Event-B; **U** => UML-B (Case 1 & 2); **E** => Event-B (Case 1 & 2)

FIGURE 5.11: The Rate of Scoring distribution for Question 4 (Unit: marks/min)

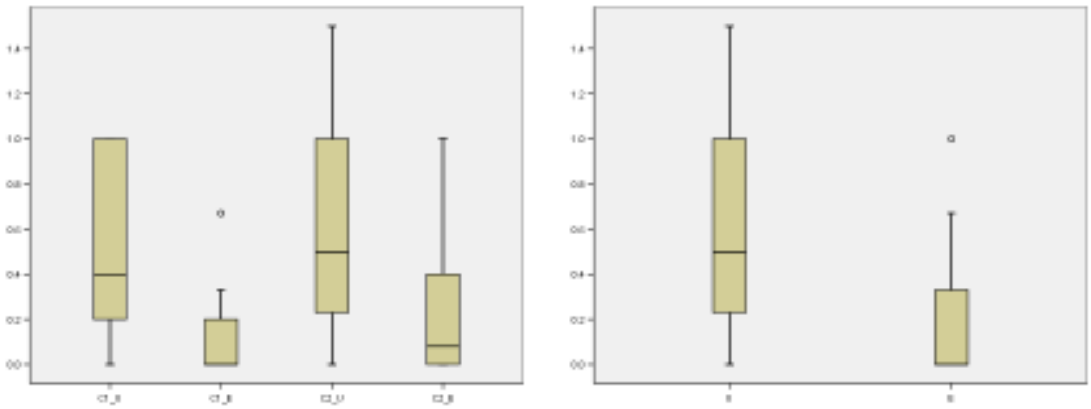
	Min	1st Quarter	Mean	Median	3rd Quarter	Max	Std Dev	N
Case 1: UML-B	0.00 (1)	0.27	0.49	0.45	0.65	1.25	0.34	14 (4)
Case 1: Event-B	0.00 (2)	0.23	0.51	0.40	0.88	1.00	0.37	15 (3)
Case 2: UML-B	0.25	0.59	0.92	1.00	1.13	2.00	0.46	15 (3)
Case 2: Event-B	0.00 (1)	0.40	0.77	0.67	1.00	1.50	0.42	17 (1)
UML-B	0.00 (1)	0.33	0.71	0.60	1.00	2.00	0.45	29
Event-B	0.00 (3)	0.33	0.65	0.67	1.00	1.50	0.41	32



Note: **C1_U** => Case 1:UML-B; **C1_E** => Case 1:Event-B; **C2_U** => Case 2:UML-B; **C2_E** => Case 2: Event-B; **U** => UML-B (Case 1 & 2); **E** => Event-B (Case 1 & 2)

FIGURE 5.12: The Rate of Scoring distribution for Question 5 (Unit: marks/min)

	Min	1st Quarter	Mean	Median	3rd Quarter	Max	Std Dev	N
Case 1: UML-B	0.00 (2)	0.20	0.50	0.40	1.00	1.00	0.42	9 (9)
Case 1: Event-B	0.00 (6)	0.00	0.13	0.00	0.20	0.67	0.23	9 (9)
Case 2: UML-B	0.00 (1)	0.25	0.60	0.50	1.00	1.50	0.45	12 (6)
Case 2: Event-B	0.00 (7)	0.00	0.22	0.04	0.36	1.00	0.31	14 (4)
UML-B	0.00 (3)	0.23	0.56	0.50	1.00	1.50	0.43	21
Event-B	0.00 (13)	0.00	0.18	0.00	0.29	1.00	0.28	23

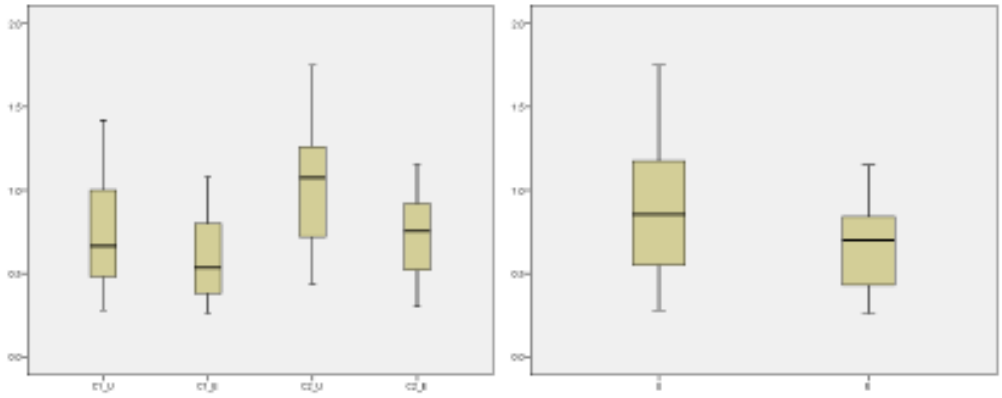


Note: **C1_U** => Case 1:UML-B; **C1_E** => Case 1:Event-B; **C2_U** => Case 2:UML-B; **C2_E** => Case 2: Event-B; **U** => UML-B (Case 1 & 2); **E** => Event-B (Case 1 & 2)

FIGURE 5.13: The Rate of Scoring distribution for Question 6 (Unit: marks/min)

Besides the descriptive statistics for individual questions, the distribution of overall understanding was also identified. The Figure 5.14 below illustrates the *Rate of Scoring* distribution for overall understanding. The data were obtained by combining the total *Score* and *Time Taken* for all six questions. The description for the columns and rows are similar to the ones explained above.

	Min	1st Quarter	Mean	Median	3rd Quarter	Max	Std Dev	N
Case 1: UML-B	0.28	0.49	0.72	0.67	0.98	1.42	0.33	18
Case 1: Event-B	0.26	0.39	0.59	0.54	0.78	1.08	0.24	18
Case 2: UML-B	0.44	0.73	1.06	1.08	1.26	1.75	0.40	18
Case 2: Event-B	0.30	0.53	0.74	0.76	0.92	1.15	0.25	18
UML-B	0.28	0.55	0.89	0.86	1.16	1.75	0.40	36
Event-B	0.26	0.43	0.66	0.70	0.84	1.15	0.25	36

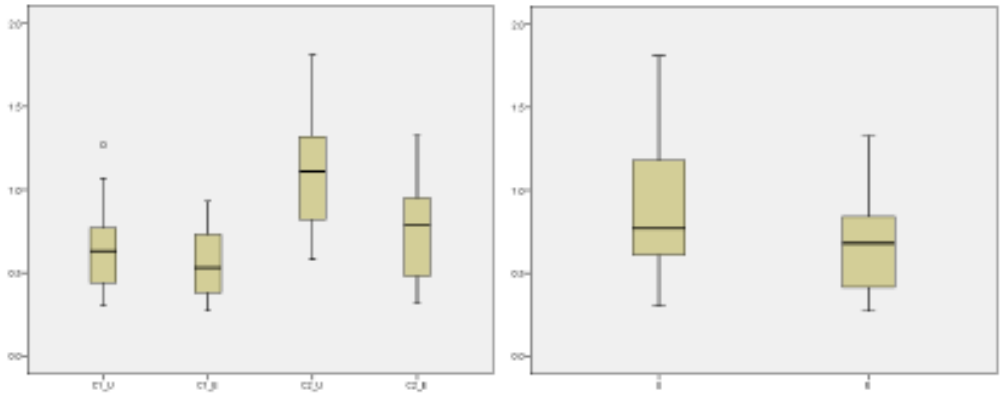


Note: **C1_U** => Case 1:UML-B; **C1_E** => Case 1:Event-B; **C2_U** => Case 2:UML-B; **C2_E** => Case 2: Event-B; **U** => UML-B (Case 1 & 2); **E** => Event-B (Case 1 & 2)

FIGURE 5.14: The Rate of Scoring distribution of overall understanding (Unit: marks/min)

The individual distributions shown above were also combined to determine specific measures of interest, namely the *Retention* and *Transfer* tests. The Figure 5.15 below shows the *Rate of Scoring* distribution for the *Retention* Test. The data were obtained by combining the total *Score* and the total *Time Taken* for *Question 1*, *Question 2* and *Question 4*. Similarly, the Figure 5.16 shows the distribution for the *Transfer* test, which was obtained by combining the total *Score* and the total *Time Taken* for *Question 3*, *Question 5* and *Question 6*.

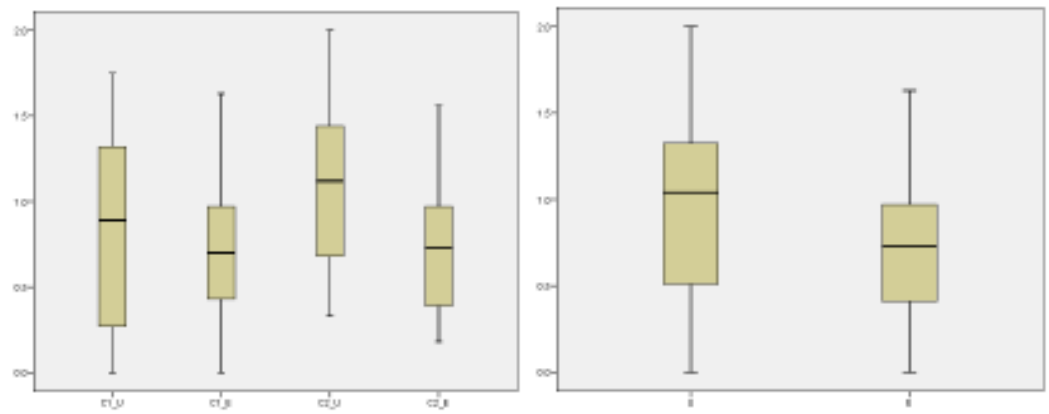
	Min	1st Quarter	Mean	Median	3rd Quarter	Max	Std Dev	N
Case 1: UML-B	0.30	0.45	0.65	0.63	0.76	1.27	0.26	18
Case 1: Event-B	0.27	0.38	0.57	0.53	0.73	0.93	0.22	17 (1)
Case 2: UML-B	0.58	0.82	1.14	1.11	1.32	1.81	0.40	17 (1)
Case 2: Event-B	0.32	0.50	0.75	0.77	0.93	1.33	0.29	18
UML-B	0.30	0.61	0.89	0.77	1.18	1.81	0.41	35
Event-B	0.27	0.42	0.66	0.68	0.84	1.33	0.27	35



Note: **C1_U** => Case 1:UML-B; **C1_E** => Case 1:Event-B; **C2_U** => Case 2:UML-B; **C2_E** => Case 2: Event-B; **U** => UML-B (Case 1 & 2); **E** => Event-B (Case 1 & 2)

FIGURE 5.15: The Rate of Scoring distribution of Retention test (Unit: marks/min)

	Min	1st Quarter	Mean	Median	3rd Quarter	Max	Std Dev	N
Case 1: UML-B	0.00 (1)	0.28	0.85	0.85	1.32	1.75	0.59	18
Case 1: Event-B	0.00 (1)	0.43	0.71	0.70	0.97	1.63	0.42	17 (1)
Case 2: UML-B	0.33	0.68	1.07	1.12	1.44	2.00	0.49	17 (1)
Case 2: Event-B	0.18	0.41	0.71	0.74	0.95	1.56	0.36	18
UML-B	0.00 (1)	0.51	0.96	1.04	1.33	2.00	0.55	35
Event-B	0.00 (1)	0.41	0.71	0.73	0.97	1.63	0.38	35



Note: **C1_U** => Case 1:UML-B; **C1_E** => Case 1:Event-B; **C2_U** => Case 2:UML-B; **C2_E** => Case 2: Event-B; **U** => UML-B (Case 1 & 2); **E** => Event-B (Case 1 & 2)

FIGURE 5.16: The Rate of Scoring distribution of Transfer test (Unit: marks/min)

From the descriptive statistics shown above, it can be seen that the means and medians of the *Rate of Scoring* on the UML-B models are higher than the Event-B

models in most of the cases. The differences are more apparent in *Question 1*, *Question 3* and *Question 6*. To recap, *Question 1* involved the description of main ideas and supporting details, *Question 3* involved the modification task and *Question 6* involved the criticism about the information presented by the models. Similar patterns can also be seen in the distributions for overall understanding, *Retention* and *Transfer* tests.

Another observation about the distributions is that the total numbers of subjects who had attempted the questions but failed to get any score are higher in the Event-B models than the UML-B models. This pattern can be seen in five out of six questions. This has caused the Event-B models' means and medians in most questions are lower than the UML-B models although their *N*s are slightly higher. This may suggest that the subjects faced some difficulty in dealing with the information presented by the Event-B models. As the UML-B and Event-B models were *informational equivalent* in each case, each subject attempted both models and subjects' variability had been randomised between the two groups, the pattern may indicate that the way information was presented by the models had an impact on the subjects' understanding.

The differences discussed above may be a reflection of true differences in the population from which the samples were taken. On the other hand, it is possible that the differences may be obtained only by chance. In order to assume that the differences obtained from the samples are true differences in the population, the standard statistical inference needs to be applied. The following paragraphs discuss the statistical inference and the hypothesis testing made on the data.

5.8.1.2. Statistical Inference and Hypothesis Testing

Prior to statistical inference and hypothesis testing, it is necessary to observe the distribution of the data to determine its normality. This is because most of the statistical testing such as t procedures or *Student's t-test* (Gossett, 1942) relies on the use of normal distributions. The normality was determined through the boxplots.

Most of the boxplots presented in the previous section indicate that the distributions are skewed. Therefore, this experiment employed the permutation tests for the statistical inference (Efron et al., 1993). The analysis was done using the S-PLUS® 7.0 for Windows-Enterprise Developer (Insightful, 2006) software.

The experiment employed a cross-over design. Thus, the data analysis considered the period effect (Senn, 2002). The detailed elaboration and justification about this type of analysis has been discussed in **Chapter 3**. In this chapter, the explanation is summarised where more attention is given on the interpretation of the data. There were two types of period effect analysis performed on the data, with and without outliers, which are explained separately below.

I. Period Effect Consideration (With Outliers)

The analysis began by obtaining the differences in the *Rate of Scoring* between the first period and the second period of both *Group Alpha* and *Group Beta*. The two-sample procedure using the permutation tests was then performed on the differences according to treatments, namely UML-B versus Event-B. The Table 5.6 below depicts the generated means and standard errors for each of the questions. It also states the t-statistics and the one-sided p-values for those questions. The t-statistics were calculated by dividing the means by the standard errors. The magnitudes of the t-statistics determined the p-values. To test the significance of the results, the p-values were assessed against the significance criterion (α). Generally, the p-values must be less than $\alpha = 0.05$ for the results to be significant. As indicated in the table, the p-values (P) of *Question 1*, *Question 3* and *Question 6* are statistically significant ($P<0.05$). These results support the descriptive statistics discussed in the previous section.

Item	Mean of Difference x 2	Standard Error (SE) x 2	t-statistics	p-value (one-sided)
Question 1	0.8877	0.2587	3.4314	0.001*
Question 2	0.0835	0.1548	0.5394	0.335
Question 3	0.9137	0.4095	2.2313	0.005*
Question 4	0.0487	0.5681	0.0857	0.483
Question 5	0.0706	0.1850	0.3816	0.339
Question 6	0.6589	0.3430	1.9210	0.020*

Note: * significant at α = 0.05

TABLE 5.6: The mean, standard error, t-statistics and p-value for each question

It is worth calculating confidence intervals only for distributions that are statistically significant. The confidence intervals determine the range of intervals about the treatment effect that can be expected in the population at a certain confidence level. To calculate the confidence intervals, the generated means and standard errors of *Question 1*, *Question 3* and *Question 6* were adjusted by dividing them by two. This was due to the doubled estimation caused by the cross-over design, which has been discussed in Experiment 1. The adjusted values are shown in the *Adjusted Mean* and *Adjusted SE* columns respectively in the Table 5.7 below. The adjusted standard errors were then multiplied with the critical values at 95% confidence level from the *t* distribution tables (Geigy 1982; Lindley et al., 1984). The critical values were determined by the software-generated degrees of freedom, as shown in the *Degree of Freedom* column. The degrees of freedom were generated based on the *No. of Trials* (*N*). Later, the values obtained from the multiplication were subtracted from the means to obtain the lowest estimated values for the treatment effect. Similarly, the values were added to the means to obtain the highest estimated values for the treatment effect. The column *95% Confidence Interval* illustrates the true treatment effect (τ) that considers the period effect at 95% confidence interval for the respective question. In essence, they are the estimated differences between the expected *Rate of Scoring* under the UML-B model and that under the Event-B model at 95% confidence interval.

Item	Adjusted Mean	Adjusted SE	No. of Trials (N)	Degree of Freedom (df)	95% Confidence Interval
Question 1	0.4439	0.1294	35	32 (~30)	$0.22 <= \tau <= 0.66$
Question 3	0.4569	0.2048	30	22	$0.11 <= \tau <= 0.81$
Question 6	0.3295	0.1715	15	12	$0.02 <= \tau <= 0.64$

Note: ~ estimated df in *t* distribution table

TABLE 5.7: The adjusted means, standard errors and 95% confidence intervals of Question 1, 3 and 6

As mentioned earlier, the experiment used the *Retention* and *Transfer* tests as the mechanism to assess subjects' understanding. These two tests determined the outcome of the learning or understanding process. In particular, they assessed whether the UML-B model could be better than the Event-B model in allowing subjects to recognise the relevant information and extend the understanding in novel situations (*Meaningful learning*) or allowing subjects to only recognise the information (*Rote learning*) or in fact, no better at all (*No learning*).

The differences in the *Rate of Scoring* for *Question 1*, *Question 2* and *Question 4* acted as the values for the *Retention* test while *Question 3*, *Question 5* and *Question 6* as the *Transfer* test. The Table 5.8 below depicts the values of these two tests. The results show that the differences in the *Rate of Scoring* for the *Retention* and *Transfer* tests between the UML-B and Event-B models are statistically significant ($P<0.05$) in favour of the alternative hypothesis. The similar result was also obtained for the *Overall* understanding ($P<0.05$), which considered all the six questions.

Test	Mean of Difference x 2	Standard Error (SE) x 2	t- statistics	p-value (one sided: alternative > null)
Retention (Q1 + Q2 + Q4)	0.4796	0.1490	3.2188	0.001**
Transfer (Q3 + Q5 + Q6)	0.5017	0.1831	2.7400	0.002**
Overall	0.4583	0.1261	3.6344	0.001*

Note: ** significant at $\alpha = 0.05$ and $\alpha = 0.025$ * significant at $\alpha = 0.05$

TABLE 5.8: The means, standard errors, t-statistics and p-values for Retention and Transfer tests and Overall understanding

In statistics, the *Bonferroni Correction* states that if *n* independent hypotheses are tested on a set of data, then the statistical significance level that should be used for each hypothesis is 1/*n* times what it would be if only one hypothesis is tested (Abdi, 2007). This is to account for the fact that *Type I error* in a set of results increases rapidly as the number of tests increases. A *Type I error* designates the mistake of rejecting the null hypothesis (H_0) when the null hypothesis is actually true. The significance criterion (α) establishes the probability of a *Type I error*. If the main hypothesis of the experiment such as stated in the **Research Question and Hypotheses** section is to be divided into two independent sub-hypotheses based on the *Retention* and *Transfer* tests, the significance criterion (α) has to be adjusted. In particular, the significance criterion $\alpha = 0.05$ has to be divided by two because of the two tests. This causes the adjusted significance criterion to be $\alpha = 0.05/2 = 0.025$.

As indicated in the Table 5.8 above, the p-value of *Retention* test and *Transfer* test is less than 0.025 ($P<0.025$). This suggests that the null hypothesis can be rejected and thus the alternative sub-hypotheses can be accepted. As these two sub-hypotheses constitute the main hypothesis, it can be concluded that the UML-B model is better than the Event-B model in fostering problem domain understanding. This claim is also supported if only the main hypothesis is tested, which in this case the overall understanding is considered. Based on the p-value of *Overall* understanding shown in the Table 5.8 above, the difference is also statistically significant ($P<0.05$). In fact, the differences are still statistically significant if a more strict significance criterion (α) is used such as $\alpha = 0.01$ ($P<0.01$). Therefore, the null hypothesis of the experiment is rejected and the following alternative hypothesis and sub-hypotheses are accepted.

Alternative Hypothesis (H_1): The UML-B model is better than the Event-B model in fostering problem domain understanding

Retention test:

$H_{1.1}$ The UML-B model is better than the Event-B model in enabling the recognition of the presented information

Transfer test:

$H_{1.2}$ The UML-B model is better than the Event-B model in enabling the construction of coherent mental models from the presented information

The significant differences in the *Rate of Scoring* in the *Retention* and *Transfer* tests shown above also implies that the UML-B model is able to promote *Meaningful learning*. Compared to the Event-B model, the UML-B model enables better understanding of problem domain where the subjects could successfully select the relevant information, organise and integrate it with prior knowledge in the working memory. Because of the differences are statistically significant, the confidence intervals calculation was pursued for the *Retention* and *Transfer* tests. The Table 5.9 below depicts the confidence intervals of *Retention* and *Transfer* tests as well as *Overall* understanding at 95%. The column *95% Confidence Interval* illustrates the estimated differences between the expected *Rate of Scoring* under the UML-B model and that under the Event-B model at 95% confidence interval.

Test	Adjusted Mean	Adjusted SE	No. of Trials (N)	Degree of Freedom (df)	95% Confidence Interval	Effect Size (Cohen's d)
Retention	0.2398	0.0745	35	33 (~30)	0.11 <= τ <= 0.37	1.12
Transfer	0.2509	0.0916	35	32 (~30)	0.10 <= τ <= 0.41	0.97
Overall	0.2292	0.0631	36	33 (~30)	0.12 <= τ <= 0.34	1.26

Note: ~ estimated df in *t* distribution table

TABLE 5.9: The adjusted means, standard errors and 95% confidence intervals of Retention and Transfer tests and Overall understanding

The results discussed so far concern statistical significance. Statistical significant results only indicate that the null hypothesis can be rejected at some level of certainty when certain statistical conditions have been satisfied. Rejecting the null hypothesis means the alternative hypothesis is accepted where the observed differences between treatments can be said to be real differences rather than due to sampling errors. Statistical significance on the other hand does not mean that the observed differences are large and thus important in practical sense. The importance or the practical significance of the results is determined through effect size measures.

The effect size measures can be measured as standardised difference between two means using *Cohen's d* index (Cohen, 1988). There are several ways to calculate *Cohen's d* index (Cohen, 1988; Rosenthal et al., 1991; Rosnow et al., 1996). One approach that is based on t-statistics (*t*) and degree of freedom (*df*) was used in this experiment ($d = 2t/\sqrt{df}$ where t = Mean/SE) (Rosenthal et al., 1991). The *Effect Size* column in the Table 5.9 above lists the effect size measures for the *Retention* and *Transfer* tests as well as the *Overall* understanding. The *Cohen's d* indices seem to suggest that the size of the effects is large (d >= 0.8) and thus important (Cohen, 1988). This means the observed treatment effects are not only statistically significant but also practically significant.

II. Period Effect Consideration (Without Outliers)

The boxplots presented in the previous section indicate that there are outliers in the data. The outliers in the data had been scrutinised to identify the possible reasons of their occurrence. One possibility was perhaps the subjects had incorrectly state the time taken as the proportion of the scores and the time taken seemed to be unbalanced. The subjects had scored twice or more as much as the time taken. For example, one subject had scored 8 marks within 4 minutes and the other 11.5 marks within 2 minutes. The scripts had been revisited to confirm the marks were correct. On the other hand, this may also reflect personal ability where the subjects were able to respond and write the answers quickly.

The analysis that excluded the outliers was performed. This enabled the results of analysis for data with and without outliers to be compared. In particular, it determined whether the conclusion of the findings would be altered if the outliers were not included. The Table 5.10 below illustrates the means, standard errors, t-statistics and one-sided p-values for the data after excluding the outliers. Similarly, the Table 5.11 depicts the values and the effect size measure for the *Retention* test.

Item	Mean of Difference x 2	Standard Error (SE) x 2	t-statistics	p-value (one-sided)
Question 1	0.8263	0.2648	3.1205	0.002*
Question 2	0.1905	0.1195	1.5941	0.058
Question 3	0.6473	0.2751	2.3530	0.009*
Question 4	-0.0698	0.4696	-0.1486	0.565
Question 5	No outliers – Same as stated in **Table 5.6**			
Question 6	0.6882	0.3762	1.8293	0.034*

Note: * significant at α = 0.05

TABLE 5.10: The mean, standard error, t-statistics and p-value for each question (without outliers)

Test	Mean of Difference x 2	Standard Error (SE) x 2	t-statistics	p-value (one sided: alternative > null)	Effect Size (Cohen's d)
Retention (Q1 + Q2 + Q4)	0.4459	0.1539	2.8973	0.002**	1.01
Transfer (Q3 + Q5 + Q6)	No outliers – Same as stated in **Table 5.8**				Same as stated in **Table 5.9**
Overall	No outliers – Same as stated in **Table 5.8**				Same as stated in **Table 5.9**

Note: ** significant at α = 0.05 and α = 0.025

TABLE 5.11: The means, standard errors, t-statistics and p-values for Retention test (without outliers)

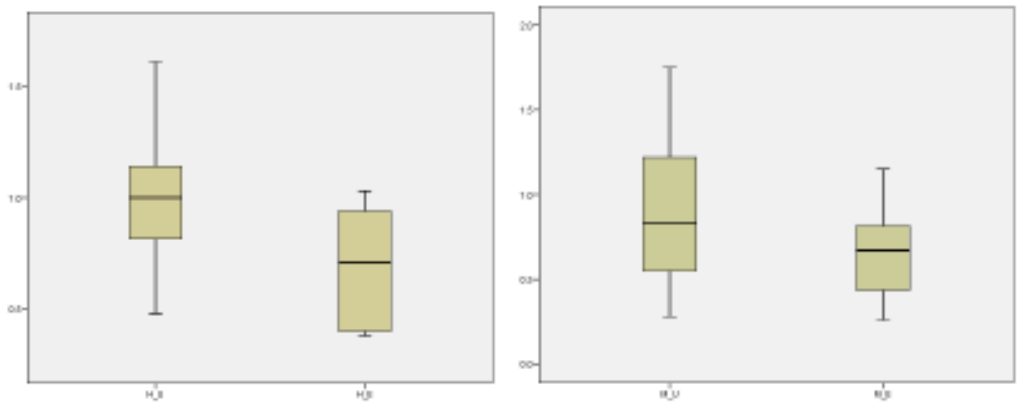
By comparing the values indicated in the Table 5.10 and Table 5.11 with the Table 5.6 and Table 5.8 respectively, it can be seen that they are pointing to the same direction. Specifically, the results of analysis for data with and without outliers suggest that the differences between the UML-B and Event-B models are statistically significant for *Question 1*, *Question 3* and *Question 6* ($P<0.05$). Moreover, the difference in the *Retention* test is also statistically and practically significant ($P<0.025$; $d>=0.8$). Therefore, the conclusion of findings described previously is still valid with or without outliers.

III. Randomised Blocking and Cross-over Design

Similar to Experiment 1, the allocation of subjects into the two groups was based on three blocks of ability: *High*, *Moderate* and *Low*. *High* ability refers to subjects who performed very well in both OO and FM courses (Grade B and above), *Moderate* ability includes subjects who performed fairly well either in OO or FM

or both (Grade C and above) and *Low* ability comprises students who performed poorly in both courses (Grade D and below). The Figure 5.17 below shows the numbers of subjects in each block for both groups. The last two columns present the means and medians of grouped overall *Rate of Scoring* for UML-B and Event-B regardless of the treatment sequence. Boxplots are also included to illustrate the distributions graphically for easy viewing and comparison. Since there were only two subjects for the *Low* ability block, no boxplot is displayed. The figure seems to indicate that the direction of treatment effect is consistent for all three ability blocks, that is, UML-B is better than Event-B. This supports the findings of the experiment although the cross-over analysis used in the experiment does not require a randomised blocking design.

Ability	Number of Subjects (Group X)	Number of Subjects (Group Y)	Total	UML-B Grouped Mean/Median	B Grouped Mean/Median
High	3 (16%)	4 (22%)	7	1.00/1.00	0.69/0.71
Moderate	14 (78%)	13 (72%)	27	0.87/0.80	0.65/0.69
Low	1 (6%)	1 (6%)	2	0.76/0.76	0.73/0.73
Total	18	18	36		



Note: **H_U** => High Ability:UML-B; **H_B** => High Ability:B; **M_U** => Moderate Ability:UML-B; **M_B** => Moderate Ability:B

FIGURE 5.17: The Rate of Scoring distribution for different ability blocks (Unit:marks/min)

5.8.2. Qualitative Measures and Analysis

In addition to the quantitative measures, there were also qualitative measures employed in the experiment. Five main aspects were measured for this purpose. The measures included the comprehension and problem solving strategies used by

the subjects to answer the questions, the subjective rating of model comprehensibility, the subjects' preference between the model notations and the subjects' personal comments on the models. The details of the questions can be found in the **Appendix C**.

5.8.2.1. Comprehension Strategies

The first qualitative measure was the direction of comprehension employed by the subjects when understanding both the UML-B and Event-B models. It aimed to discover whether the subjects employed the *Top-down* or *Bottom-up* strategy when understanding each model and whether one strategy was employed more than the other in any of the models. Similar to Experiment 1, the subjects were given specific scenarios that described the strategies, from which they had to choose one. The Figure 5.18 below shows the proportion of strategies employed in both models. The figures were based on twenty-six and twenty-eight subjects responded to the question for the UML-B and Event-B models respectively.

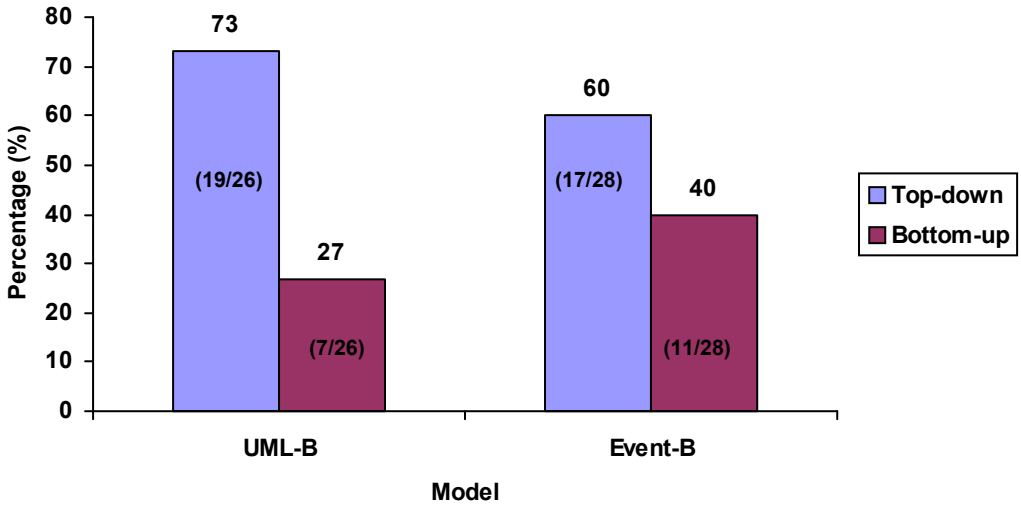


FIGURE 5.18: Direction of comprehension when understanding models (in %)

Based on the feedback shown above, most subjects employed the *Top-down* strategy when reading and understanding models. Similar trends can be seen in both models where no significant difference was found (Chi-Square test: x^2=0.45; P>0.05). This finding is similar to Experiment 1.

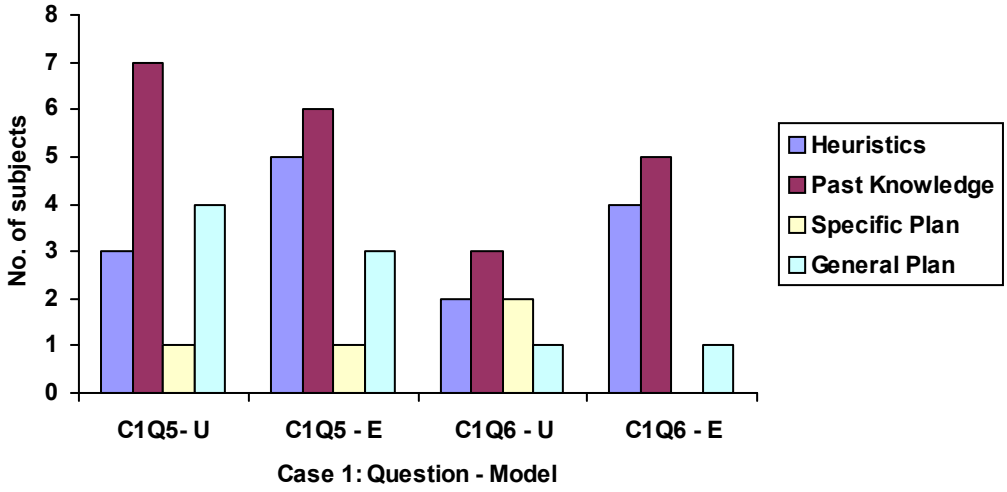
5.8.2.2. Problem Strategies

The second qualitative measure was the strategies employed by the subjects when they were confronted by a novel situation and must decide on course of action. According to cognitive psychology, there are four types of problem solving strategies that are normally used by humans, namely applying weak methods to search heuristically for a possible solution, using analogical transformation to a known solution of a similar problem, instantiating specific plans and applying general plans to reduce the problem (Ryszard et al., 1986; Smith et al., 2007). Weak methods or heuristics approach is employed when humans have little or poor knowledge about the problem domain. When a problem may be relatively familiar but there is a lack of specific plans to solve it, general plans may be applied where the problem is broken down into sub-problems. With more familiar problems, various specific plans about how to solve them have already existed in humans' minds. Thus, a specific plan can be initiated to solve the problem at hand. Finally, if humans do not have specific or general plans, they may choose a specific past experience and apply it by analogy to solve the problem. There are also instances where more than one of these approaches are applied together to solve a problem.

The Figure 5.19 below illustrates the distribution of strategies that the subjects used to solve *Question 5* and *Question 6* for *Case 1*. Similarly, the Figure 5.20 shows the distribution of strategies that the subjects used to solve *Question 5* and *Question 6* for *Case 2*. The figures were based on the subjects who responded to the questions. One observation is that the proportion of *Heuristics* is slightly higher in the Event-B models than the UML-B models. This may be only because the subjects performed on the Event-B models had poor knowledge about the problem domains by chance. In another sense however, this may also indicate that UML-B model is more able to reduce the act of "guessing" than the Event-B model.

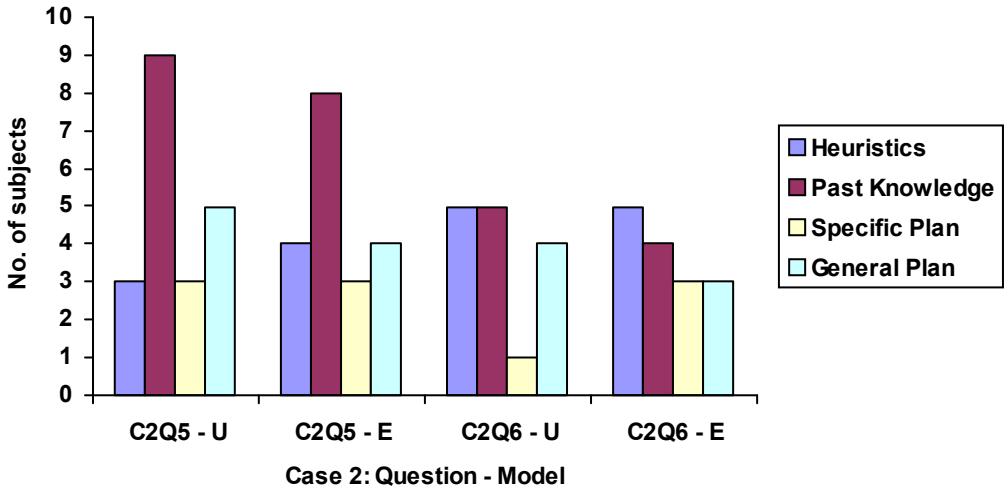
Even if the subjects had no knowledge about the problem domains, they could possibly still solve the problems by choosing a specific past experience and applying it by analogy. This could happen if the model was able to trigger the

subjects' long-term memory to find that experience. Based on the distributions, it can be seen that the proportion of subjects who used the *Past Knowledge* is a bit higher in the UML-B models than the Event-B models in most of the cases. This may indicate that although perhaps some of these subjects had little knowledge about the problems, certain aspects of the UML-B model assisted them to integrate the problems at hand with their past experience from the long-term memory.



Note: **C1Q5–U** => Case 1 Question 5 UML-B **C1Q5–E** => Case 1 Question 5 Event-B
C1Q6–U => Case 1 Question 6 UML-B **C1Q6–E** => Case 1 Question 6 Event-B

FIGURE 5.19: Problem solving strategies for Question 5 and 6 for Case 1: Auction System



Note: **C2Q5–U** => Case 2 Question 5 UML-B **C2Q5–E** => Case 2 Question 5 Event-B
C2Q6–U => Case 2 Question 6 UML-B **C2Q6–E** => Case 2 Question 6 Event-B

FIGURE 5.20: Problem solving strategies for Question 5 and 6 for Case 2: Library System

The lower number of subjects who employed the *Specific Plan* as compared to the *General Plan* may suggest that the subjects were quite familiar with the problem domains. They may have used the systems before but they were not domain experts. This had caused them to tackle the problems more cautiously. In addition, it seems that the *General Plan* was more adopted in the UML-B models than the Event-B models. Perhaps the way information is presented influences the way a problem is tackled.

5.8.2.3. Model Comprehensibility

The third qualitative measure was the subjective rating of model comprehensibility. The subjective rating was a symmetric five ordinal scales from *–2* for *Very difficult to comprehend* to *2* for *Very easy to comprehend*. The Table 5.12 below illustrates the distribution of the rating for each of the models. The figures were based on the thirty subjects who responded to the questions. As indicated in the table, most subjects rated the UML-B model as *Very easy to comprehend* while the Event-B model as *Neither difficult nor easy*.

	-2 Very difficult to comprehend	**-1 Difficult**	**0 Neither difficult nor easy**	**1 Easy**	**2 Very easy to comprehend**	**Total**	**Median**
UML-B	2	1	4	6	17	30	2
	7%	3%	13%	20%	57%	100%	
Event-B	3	6	7	5	9	30	0
	11%	21%	25%	18%	25%	100%	

TABLE 5.12: Distribution of rating on model notation comprehensibility

In comparison with Experiment 1, it seems that the rating gap between the new version of UML-B and Event-B models is more apparent than between the previous version of UML-B and B models. In fact, the rating is in the opposite direction where in this experiment the subjects seemed to rate the UML-B model as more comprehensible than the Event-B model. In Experiment 1, the B model was rated as more comprehensible than the UML-B model.

There are several possibilities for the above phenomenon. The subjective comments received in Experiment 1 indicated that the subjects were not so convinced of the efficacy of UML-B. It may be that the enhancements made in the new version of UML-B that improved the subjects' perception in this experiment. Another possibility is perhaps due to the support given prior to the experiment execution. Compared to Experiment 1, the subjects in this experiment were given a bit more supporting documentation and time to explore UML-B. This may have given the subjects more opportunity to appreciate the method. In addition, the environment may also have affected the subjects' perception of model comprehensibility. Experiment 1 was a paper-based exercise while this experiment was conducted online. It may be that the subjects could make more senses of UML-B's operability when using it online. On the other hand, it may also be possible that the gap is due to the nature of Event-B and B. Perhaps there are some internal characteristics of these two methodologies that make one model seems to be more comprehensible than the other when comparing with UML-B. This claim was supported by several subjective comments received from the subjects, which indicated that they preferred a B model to an Event-B model. This phenomenon is worth investigating in future.

In addition to the ordinal scale, the subjects were also asked several subjective questions on what aspects of the models that they found had eased and hindered the understanding. The answers were analysed together with other subjective comments received on the models, which are discussed later in this section.

5.8.2.4. Model Preference

The fourth qualitative measure involved the subjects' preference between the model notations. The Figure 5.21 below illustrates the proportion of model preferences, which were based on the twenty-eight subjects who responded to the questions. The data indicate that the UML-B model was more preferred than the Event-B model. In Experiment 1, the B model outweighed the UML-B model. It seems that in this experiment, the UML-B model was more attractive to the subjects than its counterpart.

Similar to previous question, this question also required the subjects to provide some justification of the selection. The answers were analysed together and included in the next sub-section.

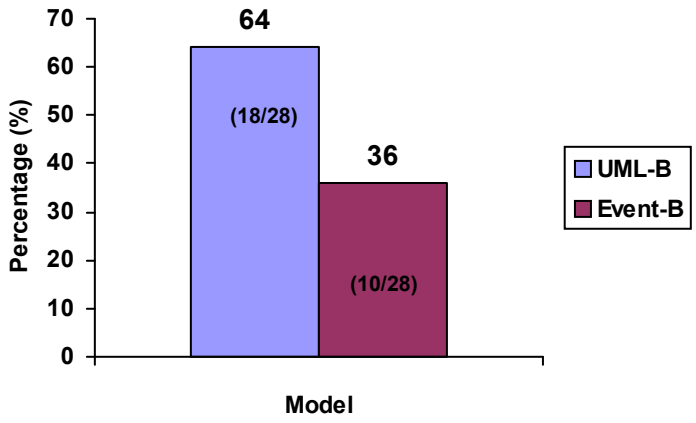


FIGURE 5.21: Preference of model (in %)

5.8.2.5. Comments on the Models

The last qualitative measure allowed the subjects to provide personal comments on the models. The comments were combined with other subjective answers given in the previous questions. The comments were categorised and conceptualised into several aspects, as shown in the Table 5.13 below.

	UML-B		**Event-B**	
	Positive	**Negative**	**Positive**	**Negative**
Approach	• Separation of static (i.e. "Context") and dynamic views (i.e. "Machine") • Top-down or black box view • Similar to existing technology (i.e. Object-oriented) • More natural to software developers	Some aspects are slightly different from UML	• Similar to programming • Sequential flow of information	The idea of having separate views (i.e. "Carrier sets", "Axioms" etc.) is not clear – "one for all view" is more preferred
Syntax	Use of visual objects	Use of "self" & "." can be confusing	Formal and less ambiguous	Too "mathematical"
Presentation	• Relationships and interactions between entities are clearly visible • Separation of views from different perspectives (i.e. UML-B, Event-B and Proving) • Use colours	• Multiple windows and diagrams • Extra boxes (e.g. machine variable) "float" on the main diagrams • Boxes with no relationships (e.g. "Context" view) look strange • Separate views can be confusing and troublesome	• "Pretty Print" – one view • Use colours and labels • Display information "plainly"	• Views other than "Pretty Print" seem useless for model viewing purposes • Lack of visualisation • Lengthy and too much text • Unnecessary navigation
Operability	• Easy and nice to use • Easy to learn • Good for viewing models	• Need extra effort to gather all views to form a meaning • Need to know where to look for what information • Can be painful for maintaining models	• Easy to view small models • Good for creating models	Can be difficult to view large models

TABLE 5.13: Subjects' subjective and personal comments on models

5.8.3. Other Findings

5.8.3.1. Absentees and Performance

The adoption of a new technology by practitioners may be influenced by the amount of training needed to use it. In Experiment 1, it has been found that even with very limited training on UML-B, one could still understand the model well. This finding is interesting especially for clients who have to read a model to validate its accuracy. These clients indeed do not require as much training as the developers, whose expertise on the technology is much more expected. Thus, it is better if the technology adopted by developers for any good reason is easy to understand by clients with limited training.

Because of the above reason, this experiment repeated the observation. The Table 5.14 below depicts the *Rate of Scoring* for subjects who were absent during the UML-B lecture. There were five subjects altogether. Four out of five subjects performed better on the UML-B model, as highlighted below. Three out of five subjects commented that they preferred the Event-B model to the UML-B model. Despite the fact that these subjects disliked and had no training on UML-B, the quantitative measures suggest that they still performed better on the UML-B model. The size of this sample is too small to perform reliable statistical significance testing. However, as two experiments produced similar patterns, the findings may be valid.

Subject	UML-B model	Event-B model	Preference
A05	0.60	0.75	UML-B
A07	**1.00**	**0.98**	**Event-B**
B01	**1.61**	**0.96**	**Event-B**
B04	**1.26**	**0.83**	**UML-B**
B06	**0.72**	**0.71**	**Event-B**

TABLE 5.14: Rate of Scoring for subjects who were absent during UML-B lecture

5.9. Threats to Validity

As this experiment was a replication, most of its threats were similar with Experiment 1. To avoid duplication, this section only discusses threats that are specifically related to this experiment.

The experiment employed a cross-over design thus it required two sets of equivalent models to be developed in each case. The equivalency was required not just between two models but also between two cases. Despite the cases must be closely equivalent, they also must be different enough to avoid any significant carry-over effect. The validity might have been affected if any of these requirements were not satisfied.

A similar set of questions was prepared for both the UML-B and Event-B models in each case. The models were *informational equivalent* and thus had quite similar levels of complexity and size. In addition, the questions in different cases had similar type and construction. Only the problem domains were different where one case involved the *Auction system* and the other was *Library system*. Different problem domains were chosen in order to reduce the carry-over effect. The *Auction system* and *Library system* were chosen as the suitable problem domains because they were not so alien to most subjects. While too expert in a problem may have caused subjects to give answers simply based on what they had already known, too alien problem domain may also have caused subjects for not being able to integrate with prior knowledge.

Both the problem domains were designed in such a way that they presented similar operations but in different contexts. Since different problem domains contained quite different operations of interest, it was impossible to make them to be exactly identical. One problem domain, for example the *Library system*, contained slightly more operations than the other. However, this was compensated by the questions' complexity level where operations in the *Auction system* were seen as a bit tricky. There were also some variations in the questions to reduce the

carry-over effect from the first session to the second besides the fact that they represented two different problem domains.

The subjects were taught formally on B for about eight hours, one hour on Event-B and one hour on UML-B. The results may have been different if the amount of training was much longer or the proportion of time spent on each method differed. However, it was believed that the allocated training time was appropriate to test the effect and was quite realistic for practitioners to adopt. The training on B was much longer than the rest because it provided the basic principles of both UML-B and Event-B. Both UML-B and Event-B training had the same number of hours and B benefited both methods. No bias was thus introduced.

The subjects might have been familiar with the systems presented by the models but were not experts. They were mainly end users who used the systems for a limited number of operations. In essence, the subjects were not expected to be expert in any of the problem domains. When individuals are expert in a domain, they already have existing knowledge structures. Rather than encode the presented information into memory, they simply adjust the existing knowledge structures to interpret the model. This means they are able to use their prior knowledge to compensate the deficiency of the models. This would confound the results of the experiment.

Several subjects had been found to face some difficulty in navigating through the models online, although training and documentation had been provided prior to the experiment. This may have affected the process of understanding. However, the invigilators had provided the necessary guidance with respect to model navigation only, without indicating the answers.

5.10. Conclusions and Future Work

This chapter has presented an experiment conducted on a UML-B model and an Event-B model. The experiment was a replication of the previous experiment that compared a UML-B model and a B model. Similar to the previous experiment, the experiment assessed the comprehensibility of the notations used in the models. The notion of comprehensibility in this experiment however was extended to include deep understanding of presented problem domains. In particular, it focused on the ability of model viewers to not only recognise the presented information but also extend the understanding in novel situations. The aim was to explore whether the UML-B model is better than the Event-B model in fostering understanding, which was measured based on the efficiency in performing the comprehension tasks.

The experiment have provided some evidence that suggests the UML-B model is better than the Event-B model in promoting meaningful understanding of presented problem domains. Meaningful understanding in this context means the UML-B model enables its viewers to actively select the relevant information, organise the selected information into meaningful mental models and integrate them with other knowledge. In addition, the results also suggest that a UML-B model is not *computationally equivalent* to an Event-B model. The UML-B model that presents information using words (textual) and pictures (graphical) is easier and quicker to understand than the Event-B model that uses only words, although they are *informationally equivalent*. This enables a UML-B model has an advantage over an Event-B model in terms of efficiency. This indicates that how information is presented can be bring an impact on how people could efficiently understand the content.

In relation with the previous experiment, the results of this experiment have supported the findings found previously. Most importantly, both experiments have demonstrated that the use of words (textual) and pictures (graphical) in presenting information is indeed more effective than the words (textual) alone. This finding brings two indications with respect to formal notation. First, the comprehensibility

of a model that uses formal notation can be significantly improved by incorporating graphical representation to its textual mathematical notation. This is particularly useful to stakeholders who have limited training but yet have to deal with such formal models. Second, if formal notation is so crucial for ensuring dependable systems and thus practitioners ought to use it, the graphical representation could make the formal notation more approachable to practitioners for quick adoption. In short, all these implications suggest that formal notation inventors should consider incorporating graphical representation into the notation to improve its accessibility. Alternatively, formal notation should be integrated with semi-formal notation to obtain the benefits of both. The high development cost associated with the development of formal models could be compensated by the improved and effective communication among stakeholders.

There are several ways in which the findings could be further explored. The experiment employed the Cognitive Theory of Multimedia Learning as the basis for its investigation. It seems that the theory is not only useful for supporting the investigation but also providing some cognitive explanation of why and how a UML-B model can be better than its counterparts in terms of understanding. As the theory consists of several principles, the experiment could be replicated by testing other principles that suit the nature of a UML-B model. For instance, the *Spatial Contiguity Principle* postulates learners learn better when corresponding words (textual) and pictures (graphical) are presented near rather than far from each other. Similarly, the *Coherence Principle* could also be tested, which posits learners learn better when extraneous words and pictures are excluded. In fact, the effectiveness of a UML-B model could be further assessed whether it affects more novices and high-spatial learners such as stated by the *Individual Difference Principle* (Mayer, 2001).

The lack of theory that supports empirical work on software comprehension has suggested that more cognitive theories should be used. Moreover, without the link from theory to hypothesis, empirical results cannot contribute to a wider body of knowledge (Kitchenham et al., 2002a). While the Cognitive Theory of Multimedia Learning has been valuable for providing some direction, more empirical work that employs other cognitive theories is highly encouraged. For example,

Cognitive Fit may be used as the underlying theory for the investigation. The *Cognitive Fit* theory states that when the cognitive processes used to act on a problem match with the way the problem is presented would lead to superior problem solving performance (Vessey, 1991). In this regard, a series of empirical work could be conducted to explore the nature of problems that could be effectively solved using the presentation format like UML-B. They could also assess whether or not the format fits with the problem domains that it should be able to solve. The findings could then be integrated to propose better ways of designing formal notation for conceptual modelling.

The experiment could be further improved by replicating the experiment using different designs, different problem domains, different variables and different samples of subject from the population. For example, it is possible to test whether the findings would be similar if the subjects are given more time for learning the method and exploring the model, a more complex problem domain is used or subjects are experts in the problem domain. Furthermore, the measurement instruments could also be improved by having more automated data collection. Several replications of the same object of study could strengthen the validity of the findings. Moreover, understanding obtained from several small-scale experiments could allow more focused and defined studies to be planned as quasi-experiments in industrial settings.

Chapter 6

Measuring the Usability of the UML-B Method – A Survey Replication

A high quality conceptual model is important to the success of system development. It reflects the ability of a system to meet the requirements of its users. The quality of a conceptual model in general is highly influenced by the notation used in the model and the environment that supports the modelling process. The notation and modelling environment transform developers' initial perception about a system to a concrete model. The model is then used as the point of reference in the subsequent development stages. Any influence that is imposed on the model by the notation and environment thus affect not only the quality of the model but also the quality of the final product.

UML-B (Snook et al., 2006) is regarded as a formal modelling method that supports the development of a conceptual model. Its notation that combines the semi-formal notation of UML (OMG, 2006) and the formal notation of B (Abrial, 1996) together with the supporting tools aim at ensuring a precise, consistent and accessible model to be developed. Since 2001, the method has evolved from a specialisation of UML to a new metamodel. The method now appears to be a UML-like formal modelling language. Despite adopting basic ideas of UML, UML-B has its own styles of graphical modelling. In fact, the formal notation used in the method has incorporated an event perspective, which targets at an incremental modelling style (Hallerstede, 2006). The tools that support the method have also evolved. Previously, the U2B translator that transforms a UML-B model to a B model resided in Rational Rose (Rational, 2000). The generated formal

model then had to be animated and verified separately by using independent B tools such as B-Toolkit (B-Core, 1999), ProB (Leuschel et al., 2003) and Click'n'Prove (Abrail etal., 2003). The translator and verification tools have now been built as plug-ins in Eclipse (Eclipse, 2007). This enables a formal model to be generated from a UML-B model, animated, analysed and verified instantly in one environment (Abrial et al., 2006).

UML-B can be regarded as a new modelling method to practitioners. Inventors in one way or another have claimed the method's strengths and opportunities in the literature (Snook et al., 2006). This has provided the practitioners with some ideas of what to expect from the method. On the other hand, its weaknesses and threats have not been explicitly articulated and immersed in technical details. This could cause some doubts among practitioners about its practicality. In particular, one concern that lies in any new modelling method is its usability. Only if the method is usable, will practitioners adopt it to gain the promised benefits.

In earlier work, two usability investigations had been conducted on the previous version of UML-B. The investigations comprised a controlled experiment and a survey. The detailed description of the investigations has been included in **Chapter 3** and **Chapter 4** respectively. The controlled experiment enabled the comprehensibility of a UML-B model to be assessed while the survey captured the nature of experience of using the method. Some of the findings of the investigations have been fed into the new version of UML-B. As the new version has become available, the investigations were replicated.

The comprehensibility of the model developed using the new version of UML-B has been explored and elaborated in **Chapter 5**. The results suggest that a UML-B model is more comprehensible than an Event-B model from the perspective of stakeholders who view the model. In particular, the UML-B model is better than its counterpart in promoting problem domain understanding. Although this finding is interesting, another usability aspect has yet to be investigated. It is important to explore whether the method itself is usable to developers who use the method during the modelling process.

This chapter presents a survey conducted on the new version of UML-B. The survey extended the investigation described in the previous chapter by focusing on the modelling process rather than its product. This survey was a replication of the survey conducted on the previous version of UML-B (Survey 1), described in **Chapter 4**. The instrument and measurement used in the survey were therefore similar to Survey 1. Specifically, it assessed the understandability, learnability, operability and attractiveness of the method's notation and supporting tools. The assessment was conducted by using a usability evaluation framework namely the Cognitive Dimensions of Notations (CD) (Green, 1989; Green et al., 1996) with several usability criteria suggested by the International Organization for Standardization (ISO) (ISO 9126-1, 2001; ISO 9126-3, 2003; ISO 9126-4, 2004).

The following section explains the technical aspects of the survey's preparation and execution. Section 6.2 and 6.3 present the results and data analysis respectively. Section 6.4 discusses the outcomes and contribution of the survey. Section 6.5 explains several threats to the validity of the results. Finally, Section 6.6 concludes the chapter with a summary of the main findings and future work.

6.1. Objectives and Methodology

6.1.1. Motivation

The objective of the survey was to capture the nature of experience of using UML-B. The survey therefore was qualitative in nature where its analysis was mainly interpretive. Most qualitative studies aim at generating theory of a phenomenon. In Survey 1, several tentative theories about the usability of the method have been proposed based on the preliminary evidence grounded in the collected data. The theories present several factors that could affect the utility of integrated methods, which combine semi-formal and formal notations such as UML-B. In this survey, the theories were evaluated and strengthened by comparing them with another set of data. The main goal was to build up the "weight of evidence" of the proposed tentative theories.

This survey was intended to achieve two purposes. First, as there are some modification and adjustment made on UML-B, the survey aimed to understand the nature of using the new version of UML-B as compared to the previous one. Second, the new version of UML-B can be regarded as another instance of integrated methods. The survey therefore attempted to discover any new factors that could affect the usability of such methods. In this sense, the earlier generated tentative theories were refined so that they could represent a more accurate description of the phenomenon.

Several comments about the usability of the new version of UML-B had been received from the subjects in the controlled experiment. For instance, the subjects were attracted to the visual aspects of the method and the idea of having separate views and perspectives. As the method resembles some technologies that are common in industry, the subjects found it as natural and intuitive. On the other hand, several subjects commented that the separate views could be confusing and troublesome. They needed extra effort to gather the related information from several views to form a meaning. They also had to be aware of where to look for the information. Thus, several subjects foresaw that it would be a bit painful for creating and maintaining the UML-B model.

To meet the objectives and confirm the validity of the above claims, the survey employed the following broad research questions. The questions were similar to Survey 1.

Do individuals who develop a UML-B model using the new version of UML-B perceive the method and model as usable (easy to understand, easy to learn, easy to operate, and attractive)?
What are the characteristics of UML-B method and model that affect their usability?

As a replication of Survey 1, the survey also had specific research questions as follows:

What are the different characteristics of the new version of UML-B as compared to the previous one?

Do the following tentative theories hold in the new version of UML-B? Is there any other theory emerging from the new data?

Theory 1: The integration of semi-formal and formal notations requires the understanding of principles and roles of both notations as well as the rules of the integration. The principles, roles and rules ought to be obvious to users.

Theory 2: The integration of semi-formal and formal notations requires strong support from the environment. Supporting tools and comprehensive documentation should be not only available but also useful, easy-to-learn and easy-to-use.

6.1.2. Materials and Approach

Similar to Survey 1, the survey instrument was developed based on the ideas proposed in the Cognitive Dimensions of Notations (CD) usability framework. The framework comprises fourteen dimensions, which acted as the variables in the survey. Moreover, the survey also adopted the grounded theory approach for the data analysis (Strauss et al., 1998). The description of CD and the justification of approach selection have been included in **Chapter 4**. In this chapter, the explanation is summarised where attention is given on the analysis and interpretation of the data.

The questions for the survey were constructed by following the proposed CD questionnaire (Blackwell et al., 2000). Despite some minor changes in wording, the questions used in the survey were similar to the previous survey except two questions. First, the question on the learnability and operability of the U2B

translator had been excluded. In the new version of UML-B, the translator is built in the modelling environment with other supporting tools. Therefore, the survey assessed the translator and the supporting tools as one rather than independent entities. Second, the question on *Premature Commitment* dimension had been adjusted to be more specific. Rather than asking whether or not the respondents *can go about any task in any order*, the question asks whether the method *allows them to go about any task in any order* or *enforces them to think ahead and make certain decisions first*. There were nineteen questions in the survey. Fourteen questions reflected the fourteen dimensions of the CD framework, four questions represented the ISO usability criteria and one question gathered suggestions for improvement. The questions were presented in random order without following a specific sequence.

Despite being a qualitative study, the questions used ordinal scales together with qualitative answers. The approach had been found to be useful in Survey 1 and thus it was repeated in this survey. Besides complementing the qualitative answers, the ordinal scales seem to "force" respondents to give some indication rather than nothing at all. This is particularly useful for respondents who are unwilling to give written textual comments. In comparison with Survey 1, the ordinal scales used in this survey were seven levels instead of five. This was to allow more specific answers. The details of the questions used in the survey can be found in the **Appendix D**.

Prior to survey questionnaire distribution, the validity and accuracy of the questions were reviewed by a focus group. There were three people involved in the process, who would use the results of the survey. The purpose of the review was to identify any missing and unnecessary questions as well as ambiguous questions and instructions.

6.1.3. Participation

Thirteen out of twenty potential participants responded to the survey. The response rate was therefore sixty-five percents. They were Masters students of Software Engineering course at the University of Southampton, who registered for

the "Critical System" course in Spring 2007 (ECS, 2007). They originated from European and Asian countries. The international students, who came from outside the United Kingdom constituted two-third of the participation. There were only two women that participated in the survey.

The participants were selected as the sample because they were taught formally on the classical B for about eight hours, one hour on Event-B and one hour on UML-B. This knowledge is necessary to develop a model using the new version of UML-B. In addition, the participants also had gone through courses on the object-oriented technology and formal methods at some points of their studies. While the subjects were familiar with UML and had been taught on B, Event-B and UML-B, they could not be considered as experts.

The participants had some practical experience of using the new version of UML-B when participating in the survey. Specifically, they were assigned a modelling task using the method within a month period. The modelling task constituted one of the coursework. They were informed to complete the survey questionnaire after completing the task. The participation was voluntary in order to adhere to the ethical policies. The participation however was highly encouraged as it provided a space for reflection on the learning prior to the examination. The participants were aware that the survey was intended for research purposes.

The survey adhered to the University's ethical policies and guidance for conducting research involving human participants (UoS, 2007). In particular, the materials and procedure used in the survey had been reviewed and approved by the University's Ethics Committee. A *Participant Information Sheet* and a *Consent Form* were enclosed together with the questionnaire. The participants were advised to read and understand the information contained in the *Participant Information Sheet* before deciding to participate. The participants were then asked to sign and submit the *Consent Form* together with the completed questionnaire.

The participants were in the final semester of their Masters course. They had reasonable amount of experience and knowledge of software development. Half of

them had some work experience for at least one year. They therefore represented closely the population under study; software developers.

6.2. Results

The survey questions assessed the UML-B model and its modelling environment. A UML-B model contains the UML-like diagrams and the Event-B notation. There are four main types of diagrams in UML-B, namely the *Package* diagram, *Context* diagram, *Class* diagram and *State* diagram. A UML-B model eventually becomes an Event-B model. The UML-B model thus includes *Machines* and *Contexts*, which are two major components in an Event-B model. The *Package* diagram is the top-level diagram that shows the structure and relationships between *Machines* and *Contexts*. *Contexts* are described in the *Context* diagram, which contains only constant data and their associated constraints. *Machines* are specified in a *Class* diagram. *State* diagrams can be attached to classes in the *Class* diagram to describe their behaviour. A formal notation, μB (micro B), is used for specifying the textual constraints and actions in the diagrams. μB is mainly the Event-B notation but with the addition of the object-oriented style dot notation and the reserved word "self". The dot notation is used to show ownership of entities such as attributes and operations by classes.

The UML-B modelling environment includes the panes for creating the diagrams and specifying the μB, the U2B translator and the verification tools that are embedded in the Eclipse environment. The translator performs syntax checking and transforms the UML-B model to an Event-B model. The verification tools allow the accuracy and consistency of the generated Event-B model to be checked.

The following sub-sections present the responses received from the respondents for each of the questions in the survey questionnaire. The first fourteen sub-sections reflect the dimensions of the CD framework while the subsequent four

sub-sections represent several usability criteria suggested by the ISO. The last sub-section is comments for further improvement.

6.2.1. Visibility and Juxtaposability

The question **(1)** of the survey assessed the ability of UML-B to allow the user to view every component of its model simultaneously or view two related components side by side at a time.

The Table 6.1 below shows the distribution of answers for question **(1)**. It can be seen that seven respondents considered the dimension as positively supported by the method, which causes the median to be "A bit easy". As the logical related components were grouped in tabs, these respondents commented that they could view different parts of the model instantly by switching the tabs. Being in an Eclipse environment, some parts of the model could be viewed side by side by dragging the windows as required. For example, it was possible to view several diagrams and their respective properties windows simultaneously. These respondents however mentioned that not all parts of the model could be seen at the same time. For instance, viewing properties of different operations or events were not permitted. In addition, different projects that were grouped in separate workspaces could not be viewed side by side unless they were grouped in one workspace, which was undesirable.

Five of the respondents regarded the dimension as negatively supported by the method. Although these respondents noted the points discussed above, they did not think it was worthwhile. For example, they found that having many diagrams displayed at the same time could be troublesome and was not helpful. Multiple windows displayed together had to be resized and rearranged to fit the computer screen. One respondent noted that even on a 17 inches monitor, it was already too small to view several windows displayed on the screen at the same time. Moreover, when the diagrams became bigger, a lot of scrolling would be needed.

One respondent thought that it was "Neither difficult nor easy" to view and compare different parts of the model due to reasons discussed above.

	-3 Very difficult	**-2 Difficult**	**-1 A bit difficult**	**0 Neither difficult nor easy**	**1 A bit easy**	**2 Easy**	**3 Very easy**	**Total**
UML-B Model & Tool	1	2	2	1	2	4	1	13
	8%	15%	15%	8%	15%	31%	8%	100%
	38%			8%	54%			100%
Median	1							

TABLE 6.1: Distribution of answers for the "Visibility and Juxtaposability" dimension

6.2.2. Viscosity

The question **(2)** of the survey assessed the degree of effort required by the user to perform a change in the UML-B model. The change in this regard includes editing the diagrams and the μB. The question required the respondents to indicate the difficulty level and state any particular changes that they found difficult or tedious to make.

The Table 6.2 below shows the distribution of answers for question **(2)**. It can be seen that eight respondents considered the task as negatively supported by the method. This causes the median to be "Difficult". These respondents found that the modelling environment was not stable which made any changes difficult. Several respondents experienced their saved data to be lost and their models were corrupted while making changes. For example, deleting associations between classes often caused the model to go haywire. Moreover, saving the changes could take some time and would be followed by many modelling errors. This made the experience not so encouraging to these respondents. One respondent highlighted that when a deletion was made on a particular element of the model, it would not cause its relationships to other elements to be deleted as well. Thus, extensive changes had to be done manually, which often caused the model to be corrupted. Two respondents highlighted that while they had to investigate errors by looking into the generated Event-B model, they were not able to change them directly as changes in the Event-B model would be overwritten by the UML-B model. They

therefore were forced to return to the UML-B model to find the corresponding location and make modifications, without even knowing whether that would eventually solve the problem. One respondent observed that not all diagrams were saved when the "Save" button was selected and the modification made directly on the diagrams would always cause problems. Furthermore, the growth of the model would require more and more time to save and verify. The respondent believed that these problems were due to the fact that the tools reside in the Eclipse environment.

The remaining respondents regarded the task as positively supported by the method. Despite the problems mentioned above, they found the task to be quite straightforward mainly because of the graphical interface.

	-3 Very difficult	**-2 Difficult**	**-1 A bit difficult**	**0 Neither difficult nor easy**	**1 A bit easy**	**2 Easy**	**3 Very easy**	**Total**
UML-B Model & Tool	4	3	1	0	3	1	1	13
	30%	23%	8%	0%	23%	8%	8%	100%
	61%			0%	39%			100%
Median	-2							

TABLE 6.2: Distribution of answers for the "Viscosity" dimension

6.2.3. Diffuseness

The question **(3)** of the survey assessed the complexity or verbosity of the notation used in UML-B to express a meaning. The notation in the method comprises the *Package*, *Context*, *Class* and *State* diagrams, and the μB. The question required the respondents to indicate how simple it was to describe what they intended in the model by using the notation.

The Table 6.3 below shows the distribution of answers for question **(3)**. It can be seen that five respondents considered the task as "Neither complex nor easy", which contributes to the median value. These respondents found that they had to think in terms of the underlying Event-B notation as well as the diagrams. They felt that thinking in both notations simultaneously could not always be achieved in

a straightforward manner. They seemed could not apply their experience and knowledge of UML, B and Event-B on the UML-B model easily. One respondent noted that it was easy to achieve what was intended using the diagrams but specifying the textual constraints and actions for the diagrams was not intuitive and thus difficult.

Five respondents regarded the task as positively supported by the method. This was mainly because of previous experience of UML and some knowledge of B and Event-B. These respondents felt that the diagrams helped them in expressing the intention, which was easier than doing it purely in Event-B. However, there was still quite a lot of formal notation that needed to be used. One respondent commented that the way the model was displayed by the modelling environment had made the task easy. The graphical interface enabled elements to be placed as intended. Having the properties displayed near to the diagrams was useful.

Two respondents felt that the task was "Very complex" because of the lack of available support for the modelling process. In particular, they found the error messages were not always understood or helpful. Because of the modelling environment was rather unstable, many unexpected behaviours had been encountered. This had caused these respondents felt uncomfortable to do modelling directly using the environment. They found much better to firstly sketch the model on papers so that they could minimise the interaction with the environment. In addition, the debugging process was also a bit complex. They needed to examine the UML-B model in order to find out why the generated Event-B model contained errors.

	-3 Very complex	**-2 Complex**	**-1 A bit complex**	**0 Neither complex nor easy**	**1 A bit easy**	**2 Easy**	**3 Very easy**	**Total**
UML-B Model	2	0	0	5	2	1	2	12
	17%	0%	0%	41%	17%	8%	17%	100%
	17%			41%	42%			100%
Median	0							

TABLE 6.3: Distribution of answers for the "Diffuseness" dimension

6.2.4. Error Proneness

The question **(4)** of the survey assessed the tendency of the notation to induce mistakes. Since the notation of UML-B involves the diagrams and the μB, the questions were divided into two parts. One was meant to assess the diagrams and the other was for the μB. The question required the respondents to indicate how easy to make mistakes when modelling the diagrams and defining the formal semantics using the μB.

The Table 6.4 below shows the distribution of answers for question **(4)**. It can be seen that nine respondents considered the diagrams as error-prone, which causes the median to be "Easy". Most of these respondents noted that it was easy to make mistakes when modelling the diagrams because of the instability and immaturity of the modelling environment. One respondent realised that the diagrams in UML-B were indeed not quite the same as the conventional UML. The same approach used when dealing with UML seemed to be not working when it was applied to UML-B. Another respondent commented that the mistakes were due to the inability of the environment to keep track of the existing naming. This caused the respondent to frequently revisit certain parts of the model to check for them. One respondent thought that better experience of the method and its modelling environment could reduce the mistakes.

Three respondents found that it was quite difficult to make mistakes when modelling the diagrams. The menu for drawing the diagrams was simple and self-explanatory. One respondent found that it was "Neither difficult nor easy" to make mistakes in diagrams.

On the other hand, nine respondents commented that it was rather easy to make mistakes when defining the formal semantics using the μB. This causes the median to be "A bit easy". Most of these respondents thought that it was easy to make mistakes because of the μB itself. Being exposed to B, they found that the μB was a bit different. There were special syntax such as "self', "**.**" and implicit keyboard entries for certain symbols, which seemed not obvious. Some confusion

had occurred in finding the correct expressions for the semantics. Moreover, it was easy to make mistakes due to the formality imposed by the μB. These respondents also highlighted that the error messages were not helpful and unpredictable. The errors that always pointed at the generated Event-B model instead of the UML-B model often confused the respondents. This caused them to ineffectively correct the errors. One respondent mentioned that as the environment hid the underlying semantics, it was difficult to grasp the overall picture of the model. Two respondents believed that the immaturity of the tool and the spatial constraints made it easy to make mistakes.

The rest of the respondents thought it was "Difficult" and "Neither difficult nor easy" to make mistakes when defining the formal semantics.

	-3 Very difficult	**-2 Difficult**	**-1 A bit difficult**	**0 Neither difficult nor easy**	**1 A bit easy**	**2 Easy**	**3 Very easy**	**Total**
UML-B Diagram	0	1	2	1	2	4	3	13
	0%	8%	15%	8%	15%	31%	23%	100%
	23%			8%	69%			100%
Median	2							
UML-B Syntax	0	1	0	3	5	2	2	13
	0%	8%	0%	23%	39%	15%	15%	100%
	8%			23%	69%			100%
Median	1							

TABLE 6.4: Distribution of answers for the "Error Proneness" dimension

6.2.5. Progressive Evaluation

The question **(5)** of the survey assessed the ability of UML-B to allow the user to evaluate his or her work in progress at any time. The question required the respondents to indicate whether or not it was possible to stop modelling at any time to check their work so far. The respondents had to state why if it was not possible.

The Table 6.5 below shows the distribution of answers for question **(5)**. It can be seen that majority of the answers is "Yes". The respondents found that whenever the sensible and coherent chunking of information had been done, they could

easily check the model. One respondent mentioned that having separate perspectives enabled the model to be partially evaluated one at a time. One respondent noted that as the environment was quite unstable, the model was saved regularly. Saving the model would allow it to be evaluated while the work was progressing.

Five respondents were not sure because they mainly were not confident with the modelling environment. Two respondents seemed to redo their work several times. One respondent mentioned that it was possible to check the syntax of the model. But, it was not really possible to check whether the partially completed model was doing as expected.

Two respondents who stated "No" believed that the checking would not be helpful if they had a partially completed model. Moreover, the saving and checking process would normally take some time to complete and the verification tools became a little unwieldy as the model grew. This hindered them to check their work regularly.

	No	Not Sure	Yes	Total
UML-B Model & Tool	2	5	6	13
	15%	39%	46%	100%

TABLE 6.5: Distribution of answers for the "Progressive Evaluation" dimension

6.2.6. Hard Mental Operations

The question **(6)** of the survey assessed the degree of mental processes required for the user to understand the notation and to keep track of what was happening. The question required the respondents to indicate whether or not they found any complex or difficult tasks to work out in their heads when modelling the UML-B model. The respondents had to state what the difficulty was, if any.

The Table 6.6 below shows the distribution of answers for question **(6)**. It can be seen that six respondents stated the answer as "Not sure". Although the UML-B

model required a bit less of formal notation, these respondents believed that it would not make it much simpler than a pure Event-B model especially for complex models. This was because the complexity mainly came from the use of formal notation itself. For instance, several respondents faced a difficulty in dealing with invariants.

The above comments were also received from the respondents who stated the answer as "Yes". One respondent mentioned that almost everything about the model and the modelling environment was difficult tasks to work out in the head. One respondent commented that the transformation of the UML-B model to an Event-B model was very confusing. Trying to figure out what had been generated made the debugging process harder. Moreover, most error messages were not actually pointed to specific places where the changes were required. Even if they did, they were not understood.

The remaining respondents stated the answer as "No".

	No	Not Sure	Yes	Total
UML-B Model	4	6	3	13
	31%	46%	23%	100%

TABLE 6.6: Distribution of answers for the "Hard Mental Operations" dimension

6.2.7. Consistency

The question **(7)** of the survey assessed whether similar semantics in the notation were presented in a similar syntactic manner. The question required the respondents to indicate whether or not they found any parts in the UML-B model that were similar in functionality but the method made them appear different. The respondents had to state what the parts were, if any.

The Table 6.7 below shows the distribution of answers for question **(7)**. It can be seen that six respondents stated the answer as "Not sure". Three respondents who stated the answer as "Yes" found some inconsistencies. For instance, there are two different ways of specifying states and actions, either in the *State* diagrams or in

the events at the *Class* diagram level. The respondents therefore were not sure which to use as both approach gave the same effect. One respondent mentioned about the confusion caused by the multiplicity. It seemed that the way multiplicity was used in the UML-B model was not quite similar with the conventional UML. One respondent was confused about the difference between the type BOOL with values TRUE & FALSE and the predicate logic Boolean having the values true and false.

The remaining respondents did not find any inconsistencies.

	No	Not Sure	Yes	Total
UML-B Model	4	6	3	13
	31%	46%	23%	100%

TABLE 6.7: Distribution of answers for the "Consistency" dimension

6.2.8. Hidden Dependencies

The question **(8)** of the survey assessed whether there was any relationship between two parts such that one of them was dependent on the other but the dependency was not fully visible. The question required the respondents to indicate whether or not they found any structure dependencies in the model. If they did, the respondents had to state how visible the structure dependencies were and what parts were involved.

The Table 6.8 below shows the distribution of answers for question **(8)**. It can be seen that one and eight respondents stated the answer as "No" and "Not sure" respectively. On the other hand, four respondents found that the structure dependencies were not visible in some parts. These respondents commented that any changes made in the generated Event-B model would not be reflected in the UML-B model. It took some time for them to realise that any changes should be made in the UML-B model rather than in the Event-B model. They expected any changes made in any model would be synchronised between the two models. Moreover, the respondents also mentioned the hidden dependencies between several diagrams such as between the *Context* diagram and the *Class* diagram. It

was not so apparent that the *Class* diagram relied on the static information in the *Context* diagram. One respondent suggested that it would be good to have the state variables in the *State* diagram to be visible on the *Class* diagram.

	No	Not Sure	Yes	Total
UML-B Model	1	8	4	13
	8%	61%	31%	100%

TABLE 6.8: Distribution of answers for the "Hidden Dependencies" dimension

6.2.9. Secondary Notation

The question **(9)** of the survey assessed the ability of the UML-B method to allow the user to provide supporting information to the model by using notation other than the official μB. The question required the respondents to indicate whether or not they could make notes or convey extra information beyond the model to themselves. The respondents had to state the possible actions, if any.

The Table 6.9 below shows the distribution of answers for question **(9)**. It can be seen that seven respondents stated the answer as "Yes". The respondents found that the *Note* facility in the modelling environment was very useful for this purpose. The rest of the respondents stated the answer as "Not sure" and "No". One of these respondents noted that although the *Note* facility was useful, it did not meet the expectation. For instance, it was only possible to state comments for a set of attributes but not an attribute. One respondent mentioned that although it was useful to have comments on the model, it would be painful when maintaining it. The respondent suggested that it would be better to have a common text file that contains the descriptions of many parts, rather than they are scattered at different parts of the model.

	No	Not Sure	Yes	Total
UML-B Model	2	4	7	13
	15%	31%	54%	100%

TABLE 6.9: Distribution of answers for the "Secondary Notation" dimension

6.2.10. Role Expressiveness

The question **(10)** of the survey assessed whether the purpose of each component in the model was obvious and the user could directly imply how it related to the whole model. The question required the respondents to indicate how easy to determine what each diagram and μB was for in the UML-B model as a whole. In addition, the question also asked whether the respondents included any component in the model without exactly knowing its purpose.

The Table 6.10 below shows the distribution of answers for question **(10)**. For the diagrams, it can be seen that seven respondents considered the task as rather easy. This causes the median to be "A bit easy". These respondents found that diagrams were easy to grasp, as they were intuitive and self-explanatory. Once the concepts were known, they could easily differentiate the role of each part of the diagrams. On the other hand, to understand the roles of different parts of the generated Event-B to the corresponding diagrams in the UML-B model required some experience. One respondent liked the idea of having separate views for static and dynamic information such as used in the *Context* diagram and *Class* diagram. Besides, the *Class* diagram was useful for viewing groups of elements and the relationships between them.

On the other hand, one respondent considered the task as "Very difficult", "Difficult" and "A bit difficult" respectively. These respondents commented that the lack of documentation and instruction on how to apply the diagrams correctly made the task difficult. Moreover, the error messages were not helpful. The respondents also mentioned about the confusion caused by multiple perspectives where at times they were not sure which perspective they should use to dissolve the errors. In addition, there was also confusion about having the *State* diagram because the respondents found that they could simply state the behaviours as *Events* in the *Class* diagram. They did not clear about what should be in the *Class* diagram and what should be in the *State* diagram. The remaining respondents considered the task as being "Neither difficult nor easy" because of the similar reasons mentioned above.

For the μB, four respondents considered the task as "Neither difficult nor easy", which contributes to the median. Lack of documentation and guidelines was the main cause. Four respondents regarded the task as quite difficult unless they understood what guards and actions were for in relation to the diagrams. One respondent mentioned that confusion could happen if the UML-B model was viewed as an Event-B. The remaining respondents found the task rather easy because they had some knowledge of B and Event-B. Besides, the grouping of elements such as guards and actions with self-explanatory labels allowed them to understand their roles. However, these respondents agreed that there were some integration rules that needed to be understood.

Five respondents commented that there were parts that they simply included in the model but they did not know the purpose. Some examples are having "Axioms" and "Theorem" in the *Context* diagram.

	-3 Very difficult	**-2 Difficult**	**-1 A bit difficult**	**0 Neither difficult nor easy**	**1 A bit easy**	**2 Easy**	**3 Very easy**	**Total**
UML-B Diagram	1	1	1	3	2	2	3	13
	8%	8%	8%	23%	15%	15%	23%	100%
	24%			23%	53%			100%
Median	1							
UML-B Syntax	0	1	3	4	2	1	2	13
	0%	8%	23%	31%	15%	8%	15%	100%
	31%			31%	38%			100%
Median	0							

TABLE 6.10: Distribution of answers for the "Role Expressiveness" dimension

6.2.11. Closeness of Mapping

The question **(11)** of the survey assessed the mapping between the notation used in the UML-B method and the problem domain. The question required the respondents to indicate how well the method allowed them to describe their problem accurately and completely as what they intended.

The Table 6.11 below shows the distribution of answers for question **(11)**. It can be seen that eight respondents regarded the mapping as positively supported. This

causes the median to be "A bit good". These respondents believed that the mapping was achieved easily because of the diagrams and its object-oriented-like approach, which was intuitive. They foresaw that the combination of UML-like diagrams and the Event-B notation would facilitate the application of the latter and make it more natural. One respondent mentioned about the support given by the environment such as *Note* facility that helped to describe aspects that were not directly indicated in the model. One respondent thought that the process would be easy if one was familiar with B and Event-B. The respondent also found that the thinking of diagrams in UML-B was not quite the same as it would be in UML. For instance, *Statechart* diagrams could be attached to other diagrams other than *Class* diagrams.

Three respondents commented that the mapping as "Neither bad nor good". These respondents agreed about the usefulness of diagrams in UML-B. They however were a bit confused when comparing it with the styles that they normally used in UML.

Two respondents considered the mapping as negatively supported. They thought they could describe the problem accurately as intended. However, nothing was certain until an error-free Event-B was produced. Unclear error messages had made the task difficult.

	-3 Very bad	**-2 Bad**	**-1 A bit bad**	**0 Neither bad nor good**	**1 A bit Good**	**2 Good**	**3 Very good**	**Total**
UML-B	0	1	1	3	2	2	4	13
Model	0%	8%	8%	23%	15%	15%	31%	100%
& Tool	16%			23%	61%			100%
Median	1							

TABLE 6.11: Distribution of answers for the "Closeness of Mapping" dimension

6.2.12. Provisionality

The question **(12)** of the survey assessed the flexibility of UML-B. The question required the respondents to indicate how well the method allowed them to play around with the model without being sure what the effect would be. The respondents were required to state which parts of the method that allowed or prevented them to do so.

The Table 6.12 below shows the distribution of answers for question **(12)**. It can be seen that seven respondents commented that the notation was not good enough for them to play around with the model. This has resulted in the median to be "A bit bad". These respondents found that the modelling environment took significant amount of time to save and verify the model whenever some changes were applied. For instance, one respondent noted that it took 15 minutes to save 3 classes on a 2GHz core duo processor. This had made them to think twice before trying out something new to see the effects. Moreover, they found that the verification process would only be worthwhile when the model was complete enough. Three respondents were a bit reluctant to play with the model because of the immaturity of the tool. They had encountered several times where the model was corrupted and thus had to redo. In addition, two respondents found that multiple error messages to be handled at one time were quite daunting.

Four respondents regarded the task as positively supported. One respondent found the verification tools were useful for the purpose. It acted as a checker of how the model was progressing and whether it behaved as what was expected. They also noted that the modelling had to be done in logical chunks. Any changes made and their effects to the model had to be understood.

Two respondents considered the task as "Neither bad nor good". These respondents mainly commented on the functionality of the environment. For instance, they could add, drag and drop elements as they liked on the diagram panes and could check the model. But, they also faced unexpected incidents such as the associations were not deleted upon deletion of a class and the model was corrupted during debugging.

	-3 Very bad	-2 Bad	-1 A bit Bad	0 Neither bad nor good	1 A bit good	2 Good	3 Very good	Total
UML-B Model & Tool	2	3	2	2	2	2	0	13
	15%	25%	15%	15%	15%	15%	0%	100%
	55%			15%	30%			100%
Median	-1							

TABLE 6.12: Distribution of answers for the "Provisionality" dimension

6.2.13. Premature Commitment

The question **(13)** of the survey assessed whether the notation used in UML-B enforced the user to make decisions prior to modelling or there was any task ordering constraints. The question required the respondents to indicate whether:

i) **the method allowed them to go about any task in any order**, or
ii) **the method enforced them to think ahead and make certain decisions first**

The Table 6.13 below shows the distribution of answers for question **(13)**. It can be seen that seven respondents commented that the method enforced them to think ahead and make certain decisions first. They found that they needed to think about the general structure of the model such as the sets, constants, classes and associations, and events to be included. Moreover, they also found that the modelling process was rather hierarchical where they tended to start with the *Package* diagram and define the constants in the *Context* diagram before dealing with the *Class* diagram and *State* diagram. One respondent believed that it was because of the "top-down" approach in UML.

Three respondents stated that the method allowed them to go about any task in any order. As far as the *Class* diagram was concerned, these respondents believed the elements could be defined independently as they liked. The ordering enforcement such as stated above was considered as logical. One respondent mentioned that when one became familiar with the method and its environment, he or she would

tend to work in a certain order. He or she would do things in chunks that could easily be checked against the generated Event-B model and be verified.

	I	ii	Total
UML-B Method	3	7	10
	30%	70%	100%

TABLE 6.13: Distribution of answers for the "Premature Commitment" dimension

6.2.14. Abstraction Gradient

The question **(14)** of the survey assessed whether the notation used in UML-B enforced any level of grouping mechanism. The question required the respondents to indicate whether the method insisted they start the modelling task by defining or grouping things before they could do anything else.

The Table 6.14 below shows the distribution of answers for question **(14)**. It can be seen that six respondents commented that they had to group certain things before they could proceed. For instance, they had to define what to be in *Machine* and *Context*, which had to be created before anything else. Moreover, the *Machine* had to be linked with the *Context*. One respondent mentioned that having to link the *Machine* and *Context* in the *Package* diagram was rather worthless. The respondents also mentioned about having the classes and appropriate static properties before including the events.

The remaining respondents stated the answer as "No" and "Not Sure" without indicating specific reasons.

	No	Not Sure	Yes	Total
UML-B Method	3	4	6	13
	23%	31%	46%	100%

TABLE 6.14: Distribution of answers for the "Abstraction Gradient" dimension

6.2.15. Learnability of UML-B

The question **(15)** of the survey assessed the learnability of UML-B. The question required the respondents to indicate how easy to learn UML-B as compared to B or Event-B. The respondents were also required to indicate any particular parts of the method that were particularly difficult to learn and understand how they worked.

The Table 6.15 below shows the distribution of answers for question **(15)**. It can be seen that six respondents found that UML-B was rather difficult to learn. The respondents mentioned about having to learn two languages at once. One respondent highlighted the confusion caused by the slight difference between the diagrams and UML, particularly the associations. Three respondents commented that the idea was easy to grasp. However, the modelling environment was not quite supportive and they had to put some effort on it. The respondents took significant amount of time to understand how to apply the method correctly using the features in the environment.

Five respondents regarded the method as quite easy to learn. This was because of the diagrams and the graphical interface. One respondent foresaw that the method has a good chance of adoption in industry. The respondent suggested the designer to reduce the use of formal notation more by replacing it with graphical elements.

Two respondents thought the method was "Neither difficult nor easy" to learn, which contributes to the median. These respondents gave similar comments as mentioned above.

	-3 Very difficult	**-2 Difficult**	**-1 A bit difficult**	**0 Neither difficult nor easy**	**1 A bit easy**	**2 Easy**	**3 Very easy**	**Total**
UML-B Model & Tool	2	2	2	2	2	0	3	13
	15%	15%	15%	15%	15%	0%	25%	100%
	45%			15%	40%			100%
Median	0							

TABLE 6.15: Distribution of answers for the learnability of UML-B

6.2.16. Usefulness of Documentation

The question **(16)** of the survey assessed the usefulness of the available manual and documentation on UML-B.

The Table 6.16 below shows the distribution of answers for question **(16)**. It can be seen that eight respondents found that the documentation on UML-B was rather useless. This causes the median to be "Useless". These respondents found the available documentation so far was not helpful enough to fully understand how to use the method properly. This included both paper and online documentation. They relied only on the tutorial materials and struggled to find more information. It was almost impossible to find the information elsewhere. Moreover, the *Help* facility was also useless. One respondent wondered how to get information about the implicit keyboard entries that were not obvious. As the available information was so little and not extensive, these respondents did not considered it as method documentation.

The remaining respondents considered the documentation as "A bit useful" and "Neither useless nor useful". The available documentation such as the tutorial materials did help them but were not sufficient for the modelling process. They expected more concrete examples.

	-3 Very useless	**-2 Useless**	**-1 A bit useless**	**0 Neither useless nor useful**	**1 A bit useful**	**2 Useful**	**3 Very useful**	**Total**
UML-B Model & Tool	5	3	0	2	3	0	0	13
	38%	23%	0%	15%	23%	0%	0%	100%
	61%			15%	23%			100%
Median	-2							

TABLE 6.16: Distribution of answers for the usefulness of UML-B documentation

6.2.17. Accessibility of UML-B

The question **(17)** of the survey assessed the accessibility of UML-B. In particular, the question required the respondents to indicate how easy to become familiar with the method and to be able to use it in their task efficiently without referring to the documentation.

The Table 6.17 below shows the distribution of answers for question **(17)**. It can be seen that six respondents found that it was "Very difficult" and "Difficult" to become familiar with the method. This was mainly because of the poor support given by the modelling environment. The error messages were not helpful and intuitive. For example, one respondent wondered why the message had to state "RPART" to indicate that there was a missing right parenthesis. The time was mainly spent on understanding what the error messages meant and rectifying the errors. The error messages were much more of a problem than a solution. More descriptive error messages were thus expected.

Five respondents commented that it was quite easy to become familiar with the method. One reason was because they were familiar with UML. They however needed more time and support on the Event-B. One respondent highlighted that it was a bit easy to become familiar because there was a tutorial session prior using the method. The respondent believed that the task would have been fairly impossible to perform if no tutorial was given.

The remaining respondents considered the task as "Neither difficult nor easy", which contributes to the median. These respondents commented that the use of UML-like concepts in the method had made the task a bit easier. On the other hand, they found that they had to refer to documentation for quite some time to really understand how the method worked.

	-3 Very difficult	-2 Difficult	-1 A bit difficult	0 Neither difficult nor easy	1 A bit easy	2 Easy	3 Very easy	Total
UML-B	4	2	0	2	2	2	1	13
	32%	15%	0%	15%	15%	15%	8%	100%
Model & Tool	47%			15%	38%			100%
Median	0							

TABLE 6.17: Distribution of answers for the accessibility of UML-B

6.2.18. Operability and Attractiveness of UML-B

The question **(18)** of the survey assessed the operability of UML-B. In particular, the question required the respondents to indicate how easy to do modelling using UML-B as compared to B and Event-B. The respondents were also required to indicate their choice in modelling, that is, which method that they would prefer to use in modelling.

The Table 6.18 below shows the distribution of answers for question **(18)**. It can be seen that six respondents found that it was quite easy to do modelling using UML-B. This was because of its graphical aspect, which was intuitive. The diagrams eased the understanding of the formal aspect of the model. In fact, the UML-B model saved their effort from having to write lines of scripts. They also found that UML-B is close to the technologies that software engineers use in industry. They believed it would be a lot of resistance in industry to adopt B and Event-B. These respondents therefore preferred UML-B.

Five respondents thought that it was quite difficult to do modelling using UML-B. The immaturity of the tool and lack of documentation were the main reason. This had caused four respondents preferred B and one respondent preferred Event-B.

The remaining respondents considered the task as “Neither difficult nor easy”, which contributes to the median. This was also due to the poor support given by the environment. One respondent stated that B was preferred.

	-3 Very difficult	-2 Difficult	-1 A bit difficult	0 Neither difficult nor easy	1 A bit easy	2 Easy	3 Very easy	Total
UML-B vs B vs Event-B	1	1	3	2	2	3	1	13
	8%	8%	23%	15%	15%	23%	8%	100%
	29%			15%	46%			100%
Median	0							

	UML-B	B	Event-B	Total
UML-B vs B vs Event-B	6	5	1	12
	50%	42%	8%	100%

TABLE 6.18: Distribution of answers for the operability and attractiveness of UML-B

6.2.19. Further Improvement

The question **(19)** of the survey provided the respondents an opportunity to raise any issue of using UML-B and its modelling environment. The respondents were also allowed to suggest any possible improvement that could be made on UML-B and its environment.

Below are some of the issues and areas for improvement highlighted by the respondents:

- Improve the performance of syntax checking and model verification. Offer some dynamic checking instead of deferring checking at model saving.
- Improve the stability of the tool and its interoperability among different platforms.
- Include other types of UML diagrams such as Use Case and Sequence diagrams.
- Provide dropdown menus that display common syntax or available choices and existing created elements for easy referencing.
- Provide more user-friendly verification interface.
- Provide automatic changes in all the respective parts of the model.
- Provide more descriptive and helpful error messages. Messages should point to the exact parts where the errors are originated or suspected.

- Provide more comprehensive documentation on the method and its environment. This includes the online help facility. The special keyboard entries for certain symbols used in the notation should be included in the documentation.

6.2.20. Other Findings

Similar to Survey 1, an informal observation had been made where the performance of the respondents when interpreting a UML-B model had been compared with their perception when developing the model. From the Table 6.19 below, it can be seen that eight out of thirteen respondents perceived UML-B as better than B/Event-B in the survey. This includes three respondents who performed better using Event-B in the experiment. On the other hand, one and three respondents who preferred Event and B respectively for modelling performed better on UML-B in the experiment. Only six respondents had been found to be consistent across the two studies, five for UML-B and one for B/Event-B. Despite the fact that both tasks were conducted online, this finding seems to suggest that there is a difference between model interpretation and creation tasks using the method and its tool.

Respondent	Experiment (Better Performance)	Survey (Better Perception)
R1	U	U
R2	E	U
R3	U	U
R4	U	B
R5	U	B
R6	E	B
R7	U	E
R8	E	U
R9	U	U
R10	U	U
R11	U	B
R12	E	U
R13	U	U

TABLE 6.19: Performance and Perception of UML-B

6.3. Analysis

Similar to Survey 1, this survey adopted the grounded theory approach for the data analysis. Besides capturing the nature of experience of using the new version of UML-B, the survey also aimed to confirm and refine the tentative theories generated in Survey 1. The theories describe the usability of integrated methods that combine semi-formal and formal notations such as UML-B. The theories comprise a set of abstract categories that are systematically connected through interrelated properties.

Two basic operations were involved in the development of the theories, namely asking questions and making comparisons. During the analysis, questions such as *what*, *why*, *how* and *when* were asked to derive the properties and their levels. As the instrument used in the survey was CD and ISO's usability criteria, the properties consist of the dimensions in the framework and several criteria in the standard. Each of the properties has a level such as *high* or *low* under specific events. Each event was compared to other events for similarities and differences across properties and levels. They were then grouped accordingly into a set of categories. Any new event emerged from the data was constantly compared with those already encountered until no new insights could be gained.

The following sub-sections list the categories and elaborate their properties. The properties were grouped into categories based on the respondents' qualitative and quantitative answers. The properties (reasoning based on CD and ISO's usability criteria) that support the statements are stated in the parentheses in the paragraphs, which link to the actual evidence described in the previous section.

6.3.1. Category 1: Model Structure and Organisation

The new version of UML-B is similar to the previous one in terms of the use of UML-like diagrams and formal notation. The diagrams are equipped with formal notation so that the characteristics and behaviours of the systems can be specified precisely. There are however several differences between both versions. In

particular, the formal notation used in the new version is Event-B while in the previous version was B. In fact, the new version of UML-B includes several other diagrams, namely *Package* and *Context* diagrams. The modelling environment is also different. In the new version, the supporting tools are embedded in one environment instead of running as individual applications.

The new version of UML-B uses Eclipse. The modelling environment has several panes to display the respective parts of a project. For example, the *Navigator* pane is similar to *Windows Explorer* (Microsoft, 2007) that displays a list of folders and the related files. In addition, there are also panes for creating and displaying diagrams, *Properties* of the diagrams, *Outline* of the model, *Problems* encountered and *Task*. The panes are organised as tabs where the users can select as necessary. As a UML-B model is eventually transformed to an Event-B model, the modelling environment also provides several perspectives. The perspectives, which are placed on the top right corner of the screen, allow the users to switch between *UML-B*, *Event-B* and also *Proving* views. The *UML-B* perspective displays the UML-B model while the *Event-B* perspective displays the generated Event-B model. The *Proving* perspective contains the automatic and interactive verification tasks for the Event-B model. The Figure 6.1 below shows an overview of UML-B model displayed on the *UML-B* perspective.

The users employ the diagrams to shows the attributes and relationships between entities and the states and transitions involved in the operations. The diagrams are created in the respective diagram panes. For instance, classes and associations are located in the *Class* diagram whereas states and transitions are in the *State* diagram. Similarly, the constant data are located in the *Context* diagram whereas the structure and relationships between the *Class* diagram and the *Context* diagram are included in the *Package* diagram. μB in the form of Event-B notation is added at different parts of the diagrams to delineate their properties, constraints and actions. This enables the UML-B model to be transformed to an Event-B model whenever it is saved. When the UML-B model is saved, the syntax checking is performed and any detected errors are displayed in the *Problems* pane.

The users can switch the perspective to view the generated Event-B model and to perform the verification task.

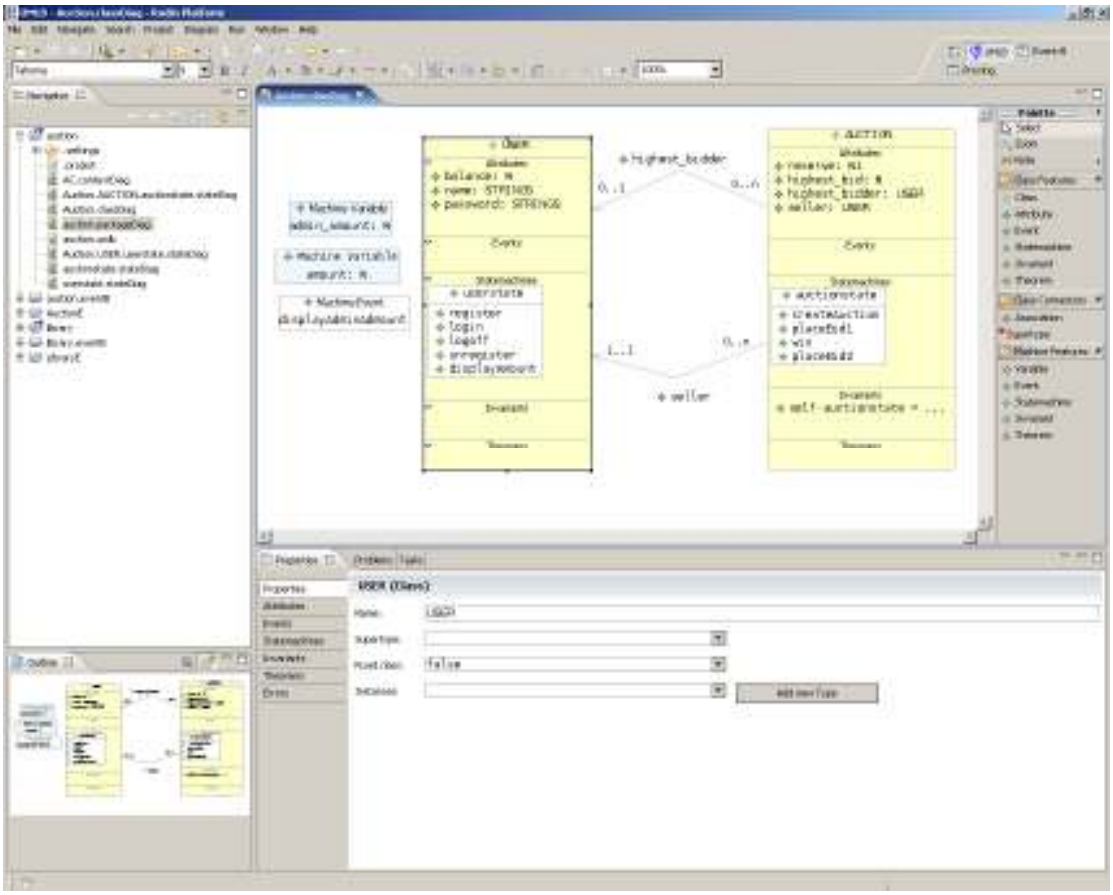


FIGURE 6.1: An Overview of UML-B modelling environment

From the above description, it can be seen that the users are dealing with multiple views and perspectives when creating a UML-B model. They are not only required to switch around different diagrams and their respective parts to specify the properties but also different perspectives. The modelling environment supports the display of multiple windows at one time by dragging the windows side by side (Property: "Visibility and Juxtaposibility" dimension). However, displaying several windows simultaneously is not always worthwhile. This is particularly true when the model is large where the users have to resize and rearrange the windows to fit the computer screen. They also have to scroll a lot in each window to view the information. Moreover, the properties, constraints and actions are scattered at different parts of the diagrams where most of them can only be seen one at a time. Thus, the information seems to be hidden. This causes the users to

overlook certain aspects of the model and prone to errors (Property: "Error Proneness" dimension). The Figure 6.2 below shows an example where two diagrams are displayed at the same time. Even after appropriately rearranging the windows, the diagrams are semi-visible. The users thus have to scroll one diagram at a time to view it entirely. In fact, the *Properties* pane can only display the information of one entity at a time. For instance in this case, only the properties of the *register* transition in the *State* diagram can be displayed.

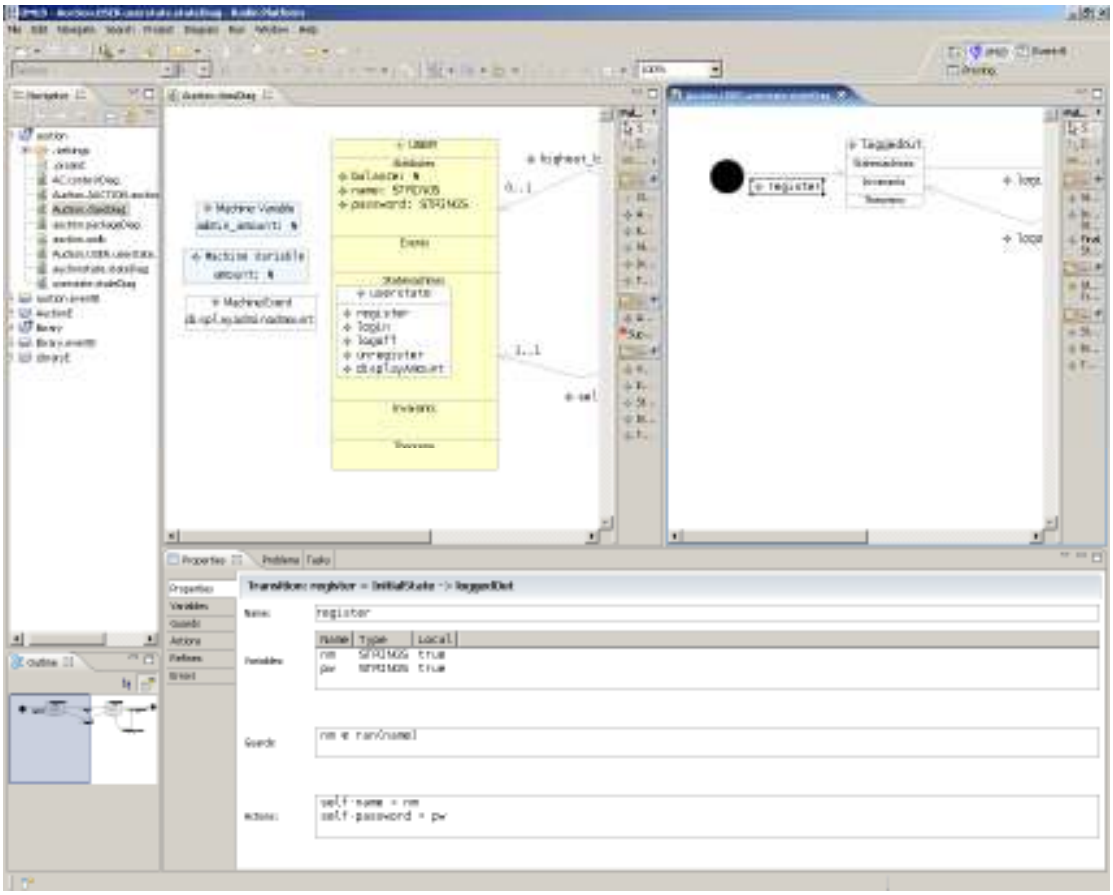


FIGURE 6.2: Visibility of multiple diagrams

The different perspectives allow the users to segregate the modelling task and check one piece of work at a time (Property: "Progressive Evaluation" dimension). In the *UML-B* perspective, the users concentrate on creating the diagrams, specifying the properties, constrains and actions, and syntax-checking the model. The users can then view the generated Event-B model in the *Event-B* perspective and verify it in the *Proving* perspective. The process however is not

straightforward when changes need to be made on the model. It is not obvious whether the changes should be made on the UML-B model or on the generated Event-B (Property: "Hidden Dependencies" dimension). The users tend to change the Event-B model directly as it is the end product that actually matters. Moreover, most error messages are pointed towards the generated Event-B model rather than the UML-B model. The environment however seems to force the users to return to the UML-B model to make changes. The users thus have to backtrack and know exactly the changes required in the UML-B model that can resolve the errors on the Event-B model (Property: "Viscosity" dimension). In a sense, this requires the users to understand the mapping between the UML-B model and the generated Event-B model, which is not easy to grasp instantly (Property: "Role Expressiveness" dimension; "Hard Mental Operations" dimension). In addition, interacting with the model using multiple perspectives also seems to confuse the users about their actual purposes (Property: "Role Expressiveness" dimension).

A UML-B model is indeed formal, albeit contains diagrams. It eventually becomes an Event-B model at the end of the process. Similar to any other formal models, there are some task ordering and restrictions of how it should be developed (Property: "Premature Commitment" dimension; "Abstraction Management" dimension). In particular, the users need to have a general overview of the model before going into details. Some parts of the model have to be defined and grouped before using them in other parts. Furthermore, the model must contain sufficient information that is organised in a logical way before it can be verified. (Property: "Progressive Evaluation" dimension; "Provisionality" dimension). Otherwise, the verification process can be ineffective and not helpful.

These findings support the subjective comments received from the controlled experiment. Several subjects commented that having separate views and perspectives could be confusing and troublesome. Thus, it would be a bit painful for creating and maintaining a UML-B model. The findings also confirm the findings of Survey 1 where the structure of the model, the organisation of its information and how they are displayed on the screen could affect the method accessibility. It is believed that integrated methods such as UML-B often involve the use of multiple diagrams that carry scattered information. As the methods

constitute several technologies, they can also have multiple perspectives. The modelling environment therefore has to allow the switching and viewing different parts of the model as conveniently as possible.

6.3.2 Category 2: Availability, Usefulness and Applicability of Supporting Tools

The modelling environment of the new version of UML-B has been regarded as supportive in some aspects (Property: "Secondary Notation" dimension; "Visibility and Juxtaposibility" dimension). The graphical interface is intuitive and similar to industrial technologies, which helps the users to grasp the idea instantly (Property: "Learnability of UML-B"). Moreover, the ability of the environment to generate a formal model has attracted the users to use the method (Property: "Operability and attractiveness of UML-B"). The automatic transformation saves some effort from modelling a formal model from scratch. Besides, the grouping of elements with self-explanatory labels and colours helps the users to recognise the roles of various parts (Property: "Role Expressiveness" dimension) and to specify elements as intended (Property: "Diffuseness" dimension).

As a newly developed technology, the modelling environment is fairly unstable. Several unexpected situations have occurred such as corrupted models and data loss. The modelling process is thus painful and frustrating (Property: "Viscosity" dimension). It discourages the users to play around with ideas (Property: "Provisionality" dimension) and restricts the way they specify the model (Property: "Diffuseness" dimension). The support provided by the syntax checker and verification tools is also poor and unreliable. In particular, the generated error messages are not always helpful and descriptive. As the errors messages cannot be easily understood, the users tend to make more errors on the model (Property: "Error Proneness" dimension). In fact, most errors are not directed to the exact positions where the problems originate. This burden the users with unnecessary mental operations to identify what goes wrong (Property: "Hard Mental Operations" dimension). Moreover, the verification tools are useful only when the model is descriptive enough. The tools in fact could become unmanageable when

the model grows. This hinders the users from checking the model as they are progressing (Property: "Progressive Evaluation" dimension).

The modelling environment is also incomplete where some expected features seem to be absent (Property: "Future Improvement"). For example, it does not support some changes automatically and the structure dependencies between several parts are not fully visible (Property: "Hidden Dependencies" dimension). This causes the modification process to be unnecessarily tedious (Property: "Viscosity" dimension). The users thus tend to make simple mistakes (Property: "Error Proneness" dimension) and are overwhelmed with numerous error messages at later stages. Having to handle many error messages that are barely understood seems to inhibit the users from trying different ideas (Property: "Provisionality" dimension).

As a method that strictly imposes model checking and verification, its environment is equipped with syntax checker and verification tools that are running behind the interface. The checking and verification process is executed each time when the UML-B model is saved. The tools are recognised as being useful for generating an accurate model. However, their slow performance seems to be intolerable. This causes inefficiency in making changes to the model (Property: "Viscosity" dimension). It also affects the users' willingness to progressively check and play with the model (Property: "Provisionality" dimension; "Progressive Evaluation").

As a new technology that is insufficiently supported by a reliable modelling environment, it is difficult for the users to become familiar with UML-B without documentation and training (Property: "Accessibility of UML-B"). In fact, the method currently lacks useful documentation (Property: "Usefulness of Documentation"). The users have to put a significant amount of effort and time to understand how to use the method and its environment (Property: "Learnability of UML-B"). Due to lack of guidance of how to apply the method properly using the environment, the users lose confidence in using the method (Property: "Operability and attractiveness of UML-B").

These findings confirm one of the tentative theories generated in Survey 1. In particular, the integrated methods such as UML-B require strong support from the environment in various aspects. The support must not only be available but also reliable and meet its purpose. In comparison with the previous version of UML-B, the current environment has been more useful. For instance, it has introduced a more seamless modelling environment, which integrates the necessary tools under one unit. In fact, the graphical interface has been better than the previous one. However, there are several aspects that should be improved in order to increase its utility (Property: "Future Improvement").

6.3.3. Category 3: Learnability and Applicability of Notations

The new version of UML-B has similar notation as in the previous one. Specifically, it uses diagrams that are equipped with formal notation so that they can be transformed to a formal model. There is however a few slight changes. UML-B is now a UML-like formal modelling language rather than a specialisation of UML. In fact, it uses Event-B as the formal notation, which is a new generation of B. The use of diagrams in the new version is thus quite different from the conventional UML diagrams. This is due to the mapping rules between the diagrams and the Event-B notation that UML-B needs to satisfy.

Previous experience of UML is useful to the users for applying the method (Property: "Accessibility of UML-B"). On the other hand, it also causes the users some difficulty in expressing the ideas accurately (Property: "Closeness of Mapping" dimension). As the diagrams look like UML diagrams, the users tend to apply the same approach that they normally use in the conventional UML to the UML-B model. The approach is not always applicable. Thus, mistakes due to confusion are often made (Property: "Error Proneness" dimension). Lack of understanding of how UML-B actually works is the main cause (Property: "Usefulness of Documentation"). A similar scenario was also found in Survey 1.

It seems that the users need to know UML and how the notation is manipulated to suit UML-B. The users also have to be familiar with Event-B, which is evolved from B. This means there are at least two notations that the users need to learn and

grasp (Property: "Learnability of UML-B"). Rather than thinking the notations individually, the users in fact have to integrate and harmonise two different styles of modelling. Thinking several notations simultaneously to express an idea is not straightforward (Property: "Diffuseness" dimension). Moreover, the users also need to understand how the UML-B model is transformed to an Event-B model (Property: "Hard Mental Operations" dimension). This includes the roles of different parts in the UML-B model that enable the transformation as well as in the Event-B model as the result of the transformation (Property: "Role Expressiveness" dimension). The understanding is particularly required when the users have to backtrack the errors from the Event-B model to the UML-B model. To ease the process, a comprehensive documentation is highly needed (Property: "Usefulness of Documentation"). Intensive training may also be useful for the purpose. The documentation should cover the practical aspects of the method with meaningful examples of application.

UML-B is seen as more approachable than its counterparts (Property: "Operability and attractiveness of UML-B"). Having diagrams together with formal notation is regarded as useful. The diagrams allow the users to describe the ideas a bit easier compared to using the formal notation alone (Property: "Diffuseness" dimension). Modelling the diagrams is straightforward as their purposes are quite obvious (Property: "Role Expressiveness" dimension). However, specifying the diagrams with properties, constraints and actions is quite complex and error-prone (Property: "Diffuseness" dimension; "Hard Mental Operations" dimension; "Error Proneness" dimension). This is due to the formality imposed by the formal notation. The users need more time and support to become familiar with the formal notation (Property: "Accessibility of UML-B").

There are some parts in the notation that appear to be redundant (Property: "Consistency" dimension). For instance, the users can specify the operations as *Events* in the *Class* diagram or as transitions in the *State* diagram. While providing some flexibility in modelling, this may also cause uncertainty especially to new users. The users are not clear which to use at what situation and how much information should be included in one diagram over the other (Property: "Role Expressiveness" dimension). In addition, the role of diagrams such as the *Package*

and *Context* diagram is so simple and thus not worth having. Moreover, the structure dependencies between diagrams are not visible in both diagrams (Property: "Hidden Dependencies" dimension). For instance, there is no indication in the *Class* diagram that the constant data reside in the *Context* diagram and vice versa.

These findings are similar with the findings found in Survey 1 and confirm one of the proposed tentative theories. Combining diagrams with formal notation could help users to particularly approach the latter. However, the users of integrated methods such as UML-B have to understand the principles of both notations. The users also have to be aware of the adjustments made in the individual notations to suit the integration. This means the individual notations may not behave the same way as they do originally. The understanding could be eased if the roles of the notations are obvious and distinct. As such methods carry the load of two notations and modelling styles, a simple integration approach is highly desirable.

6.4. Discussion

The main objective of the surveys conducted on UML-B was to generate a set of theories that describe the usability of such methods. In addition, the surveys also aimed to propose a guideline for designing integrated methods such as UML-B. The following sub-sections discuss the theories and guidelines respectively.

6.4.1. Theory Generation

The aim of this survey and Survey 1 described in **Chapter 4** was to generate theories about the usability of methods that integrate the use of semi-formal and formal notations. UML-B is one instance of such methods. UML-B in itself has undergone some changes since its first invention. The surveys investigated the two variations of UML-B to explore whether the theories about its usability would vary under different environments and conditions.

The analysis of both surveys seems to reveal similar patterns. The usability of such methods depends on the capacity of users to grasp the underlying principles of both participating notations. The methods are therefore more attractive to users who are already familiar with the concepts. Both notations have their own characteristics and styles, which can be quite different from each other. To meet the objectives of integration, the notations' individual characteristics are manipulated through the use of specific integration rules. Rather than thinking the notations individually, users thus need to understand how both notations work together under the rules. This includes the roles that both notations play in the integration. The understanding is difficult for users to acquire instantly. The methods must define and articulate clearly the principles and roles of the notations. The rules of integration should also be specified succinctly where inconsistencies and ambiguities should be avoided as much as possible. Below are some of the comments received from the respondents that support this claim.

Respondent 1:

"Provided that I already have experience on B and UML, it takes significant time to understand how to use correctly the method and its tool and understand the purpose of its features. Since I have some experience with UML, I could understand the basic concepts of the diagrams."

Respondent 2:

"To a person such as myself that has been in so much contact with UML, it was much easier to produce a coherent and more complicated model resorting to my UML basis than it would have been with just B knowledge. I applaud the idea of merging both concepts. In general, I could say that I would always prefer to work with UML-B than just classis B/Event-B. But, you are learning 2 languages not just one."

Respondent 3:

"Errors occurred because I had identical attribute names in separate classes. That caused some confusion because I had forgotten that the model would be converted into Event-B where names must be unique. This confusion was due to practices in standard UML where the user can set public or private attributes to control interference between classes, and names are bound to the scope in which they are defined."

Respondent 5:

"One the one hand, you need to think in terms of the underlying B/Event-B, on the other hand there are graphical elements that resemble UML model elements. The switch between both worlds cannot always be achieved in a straightforward way."

Respondent 8:

"If you are familiar with UML and B/Event-B, the purpose of the diagrams and syntax are deducible. However, there are some UML specific adaptations. But, there is no documentation available in the tool itself."

Respondent 9:

"When I see classes in UML-B, I always apply my experience of UML classes in UML-B and think about classes and objects. However, it does not work in most cases. I cannot apply entirely my knowledge of UML or B/Event-B on UML-B. It seems to me that UML-B has some adjustments on how to use the UML diagrams and B syntax."

Respondent 10:

"What is the difference between Logical TRUE and BOOLEAN true? Do we need to always have statecharts? I didn't know that <MemberOf> operator is produced by the keystroke – all these are not explicit. The notation is slightly different from B/Event-B which has caused me problems in finding the correct expressions for the semantics."

Both surveys also suggest that such methods should be equipped with strong supporting tools. The supporting tools not only encompass the internal devices that support the modelling process but also the external devices such as documentation and training. The data from the second survey particularly have shown the impact of poor supporting tools on the usability of such methods. Although the methods are able to attract users who opt into formal modelling, the inadequate support from the environment has lessened the interest. The tools should guide users through the learning and development process. They also should reduce the complexity of dealing with multiple notations and modelling

elements. As the environment of such methods normally involves more than one interacting elements, the tools should not only be effective but also efficient. Below are some of the comments received from the respondents that support this claim.

Respondent 3:

"The method combines the power and simplicity of UML and provides you a better environment to design your B models. It helps you to visualise things and gives you better understanding of what is going on. But, I want to point out that the tool has too many bugs and sometimes it is really complicated to use the tool. Its error reporting system is helpless. The tool is lacking from error reporting and documentation."

Respondent 4:

"It must be said that the experience with the tool has been a ghastly one. Inconsistencies in the tool turned it at some points into a downright frustrating experience. There are three fundamental flaws – lack of response of the graphic environment to change, use a lot of memory space and the error messaging is simply not up to any desirable level."

Respondent 6:

"There are many bugs, making it stressful to use. Errors shown are not always easy to understand. Due to lack of documentation, errors are totally ambiguous and hardly resolvable. What is the meaning of this message : Error while running tool (Machine Static Checker)??? Building a model consisting of only three classes sometimes takes up to 15 minutes. Thus, I think twice before saving the model to view the effects of their changes."

Respondent 7:

"The method needs a sophisticated tool, able to handle input errors correctly and produce detailed and straightforward error messages. Users should not be forced to refer to a lower abstraction level to understand what they did wrong. The documentation and training are necessary."

Respondent 13:

"The manual does not give enough detail of how to use UML-B. I did not find any adequate documentation online. The interface is flexible and convenient but the provided documentation is not adequate."

In Survey 1, model structure and organisation has been identified as one of the potential factors that might affect the usability of integrated methods such as UML-B. However, it has not been suggested explicitly in the tentative theories. This is because the survey was the first investigation of the method in a specific environment. Besides, the data that supported the factor were also limited. It may be that the phenomenon could only be observed in that particular modelling environment and therefore could not be safely generalised. In this second survey, a different modelling environment was used. Despite some variations and

improvements made in the environment, it seems that the structure of the model, the organisation of its information and how the information is displayed on the screen have an impact on the method accessibility. Users with high spatial ability, who are able to conceptualise the spatial relations between different parts, are believed to be at an advantage when dealing with such models. Below are some of the comments received from the respondents that support this claim.

Respondent 2:

"I have to investigate errors by looking into the generated Event-B but I am not able to change them directly in this view. Instead, I am forced to switch back to UML-B view, find the corresponding location in the model and make modifications, not knowing whether this would eventually solve the problem."

"Diagrams get very big, which means a lot of scrolling will be needed. Resizing all the diagrams to have multiple windows opened and viewed together is not worth the effort."

Respondent 11:

"I have to transfer tabs when I want to see other views. The tab interface is useful. But, running 2 different workspaces is not possible to view two projects side by side. They have to be in the same workspace folder, which is undesirable."

Respondent 8:

"I suppose that Eclipse has facilities to put windows side by side, but even on a 17 inch laptop the windows are already too small without having several of them on the screen at the same time."

Respondent 9:

"There is a large amount of overlaps between the windows where you enter information and ones which reflect generated code (which you should not alter)."

Respondent 12:

"The different parts of the model are grouped in tabs, making easy to switch between them. Also, the different perspectives provided are easy to be switched."

Respondent 13:

"Fairly easy - The properties are opened in a separate panel, the diagrams in a separate tab in the main panel. But, viewing items at the same time is possible only for certain cases."

In short, the tentative theories of the usability of integrated methods that combine semi-formal and formal notations based on data from two surveys are as follows. The tentative theories generated in **Chapter 4** are refined so that they explicitly describe the phenomenon. This provides some guidance for more specific investigations to be planned in future studies. The categories that contribute to the formulation of the theories are stated in the parentheses:

Theory 1: The integration of semi-formal and formal notations requires deep understanding (**Category 3**).

- Understanding of the underlying principles of each participating notation
- Understanding of the roles that each participating notation plays in the integration
- Understanding of the rules of integration, which manipulate the notations' individual characteristics to meet the objectives
- Principles, roles and rules of integration should be obvious and easy-to-understand

Theory 2: The integration of semi-formal and formal notations requires strong support from the environment (**Category 2**).

- Support from the internal devices that accompany the method, which should be functionally effective and efficient
- Support from the external devices such as documentation and training, which should be comprehensive and useful
- Supporting tools should be easy-to-learn and easy-to-use to facilitate understanding and application

Theory 3: The integration of semi-formal and formal notations requires well-organised models and high spatial ability (**Category 1**).

- Related information should be presented near rather than far from each other
- Roles of different parts of the model should be obvious and unique where no extraneous or redundant functionality
- Users should be able to conceptualise the spatial relations between different parts of the model

6.4.2. Design Profile

In Survey 1, the goals for designing integrated methods such as UML-B were identified. The goals were proposed based on the nature of semi-formal and formal notations, the motivation behind the integration and the common user activities involved in using the methods: *Exploratory Design* and *Modification*. The goals are in the form of CD profile where each of the fourteen dimensions has a specific desired level. The Table 6.20 below illustrates the proposed CD profile, which details have been explained in **Chapter 4**. The profile aims at providing method designers some guidelines when planning such methods.

Based on the results obtained from the two surveys conducted on UML-B, there are three major elements that could potentially influence the usability of integrated methods. First, the notation used in the methods, which combines the use of semi-formal (graphical) and formal (textual) notations. Second, the structure and organisation of the information contained in the methods' model. Third, the accompanying tools that support the user activities using the methods. In general, the *Notation* determines the syntactic property, the *Model* represents the structural property and the *Tool* is the operational property.

The proposed CD profile in **Chapter 4** is extended to include the three elements. To achieve a desired level for a specific dimension, at least one of these elements is involved. The elements act as the focus of design. The Table 6.20 below depicts the elements with respect to each dimension. For instance, to achieve *Low*

Abstraction Gradient, the *Notation* is the centre of attention. Method designers are suggested to put more thoughts on the syntactic property to meet the desired level. On the other hand, the *High Role-expressiveness* requires the method designers to consider not only the syntactic property but also the structural and operational properties of the methods. There are dimensions that require designers to focus on the aspects other than the notation, namely model and tool. For example, to achieve *Low Viscosity*, the structure of the information in the model should be easy to change and the tool should be able to ease the process. The evidence that support this proposal can be found in the respective dimensions in the **Results** section.

Dimension	Desired Level	Notation	Model	Tool
Abstraction Gradient	Low*	√	√	
Closeness of Mapping	High*	√		
Consistency	High**	√		
Diffuseness	Moderate (instead of Low)*	√		
Error-proneness	Low*	√	√	√
Hard Mental Operations	Low*	√	√	√
Hidden Dependencies	Low		√	√
Premature Commitment	Low*	√	√	
Progressive Evaluation	High**	√	√	√
Provisionality	High			√
Role-expressiveness	High*	√	√	√
Secondary Notation	Moderate (instead of High)			√
Viscosity	Low		√	√
Visibility/Juxtaposibility	High		√	√

Note: High – to increase; Low – to reduce; Moderate – possible trade-off; * – Semi-formal notations support formal notations to achieve the desired level (otherwise, the level will be opposite); ** – Formal notations support semi-formal notations to achieve the desired level (otherwise, the level will be opposite).

TABLE 6.20: The proposed Cognitive Dimensions profile and focus of design for designing integrated methods (combine semi-formal and formal notations)

This profile has been suggested based on the findings of two surveys conducted on UML-B. It may not be conclusive where they can be validated and refined further in future investigations on other instances of integrated methods. However, it provides some guidelines to better design of integrated methods such as UML-B.

6.5. Threats to Validity

As this survey was a replication, most of its threats were similar with Survey 1. To avoid duplication, this section only discusses threats that are specifically related to this survey.

Similar to Survey 1, one question seems to mislead a few respondents unintentionally. The question was for *Error Proneness* dimension. Although the problem had been envisaged before the survey execution, not many changes could be done to the question. This was because any major changes made to the question could divert the question from its actual intention. As this survey was a replication of the previous one, the changes could also make the comparison of both results a bit difficult. By having both qualitative questions together with ordinal scales however had overcome the problem.

The survey questionnaire was distributed to all twenty Masters students of Software Engineering course at the University of Southampton, who registered for the "Critical System" course. Thirteen students responded to the survey. Although the number was quite small, the response rate of sixty-five percents was considered as appropriate for an initial investigation. Brief identity screening was done on the students who did not respond. No particular pattern was identified that could have potentially biased the results.

Using an odd number of levels for the ordinal scale may have left open the possibility of non-committal responses such as "Not sure". Under these circumstances, the analysis of the data mainly relied on the qualitative answers that were gathered from various questions in the survey.

The respondents were taught formally on B for about eight hours, one hour on Event-B and one hour on UML-B. They were assigned a modelling task using UML-B within a month period. The period might have been insufficient for the respondents to fully experience the method, which had caused non-committal responses. Perhaps, the results would differ if the respondents were given more

time and training. The aim of the survey was to capture the experience of using UML-B from new users' perspective. Therefore, the allocated time frame and training were seen as adequate and realistic for the purpose.

6.6. Conclusions and Future Work

This chapter has presented a survey conducted on UML-B. The survey was a replication of the previous survey. Previously, the usability of the earlier version of UML-B was assessed whereas this survey assessed the latest version of UML-B. Similar to the previous survey, the assessment was conducted using the Cognitive Dimensions of Notations (CD) framework and the International Organization for Standardization's (ISO) usability criteria. The survey attempted to understand the nature of experience of using the new version of UML-B. In addition, it also aimed to confirm and refine the tentative theories of the usability of UML-B, which were proposed in the previous survey. It thus employed the grounded theory approach for the analysis. The analysis also enabled a design guideline to be proposed.

UML-B has been regarded as a user-friendly way of formal modelling. The visualisation of components and their relationships has made a UML-B model more intuitive and approachable than developing a textual formal model. Moreover, UML-B's graphical interface and interactive modelling environment is more oriented towards the needs of developers. The environment allows the developers to deal with visual objects at abstraction level and go into details as necessary. While being a very useful method to construct and verify a formal model, there are some improvements need to be made. In particular, the method has to improve its supporting tools and functionality as well as its model's structure. This is to help its users to face the challenge of having to interact with multiple notations and elements.

The survey mainly confirms the theories generated in the previous survey. There are three main elements that could potentially affect the usability of integrated

methods such as UML-B. They are the structure of the models, the support provided by the tools that accompany the methods and the notations used in the methods. Well-organised models with easy navigation and viewing, strong supporting tools and role-expressive notations that are easy to grasp are seen as necessary for the successful use of such methods. Due to their importance, these three elements need to be considered along with the dimensions of usability when designing integrated methods. The elements act as the focus of the design. The proposed design guideline included in this chapter covers these aspects.

The findings of both surveys could be improved further by extending the assessment to a large number of users. More experienced users from the industry could be invited to participate in the survey. Moreover, by investigating other integrated methods such as UML and Z (Dascalu et al., 2002; Martin, 2003; Kim et al., 2004), UML and Object Constraint Language (OCL) (Warmer et al., 2003) or other instances of UML and B integration (Shore et al., 1996; Sekerinski et al., 1998; Meyer et al., 1999; Ledang et al., 2002; Lano et al, 2004) could test the generated theories. This would result in better understanding of the phenomenon. In addition, each of the proposed theories could be investigated solely to capture the possible effects under varying conditions. For instance, the process of understanding the notations used, their roles and rules of integration could be investigated by using phenomenographic research approach (Akerlind, 2002). The aim of such a study is to describe the variations of experience of learning. The findings of the study could be used to formulate strategies to facilitate the learning and understanding such methods. Furthermore, future studies could also look into the proposed design guidelines and improve it further by incorporating other usability criteria.

Chapter 7

Measuring the Usability of Verification Tools

UML-B is an instance of modelling methods that integrates the use of semi-formal and formal notations (Snook et al., 2006). The rationale of such integration is to enable an accurate and precise model to be developed using notations that are more accessible to practitioners. While the accessibility of the notations is achieved through the use of the semi-formal notation, the accuracy and consistency of the model are assured through the formal notation. The formality imposed by the formal notation enables the model to be verified systematically by tools, which are designed specifically to increase model precision and consistency. In the case of UML-B, this is achieved by using the available B tools such as ProB (Leuschel et al., 2003), Atelier-B (ClearSy, 2003), B-Toolkit (B-Core, 1999) and Click'n'Prove (Abrail et al., 2003).

The development of a UML-B model can be regarded as complete only when the B tools have successfully verified the model. Therefore, the usability assessment of such a method should include not only the tools that transform its model to a formal model but also the ones that verify the generated formal model. In the new version of UML-B, the verification tools have been embedded in one environment together with the modelling editor and translator. The usability of the tools have thus indirectly been assessed in the earlier survey described in **Chapter 6**. On the other hand, it is worthwhile to explore specifically the verification tools in order to understand some aspects of their usability. After all, the main motivation of methods such as UML-B is to allow its model to be verified by such tools.

This chapter presents a survey conducted on two instances of B tools, namely ProB and B-Toolkit. The survey aimed to explore which features are necessary for verification tools to become usable. Similar to the previous surveys, the survey employed the Cognitive Dimensions of Notations (CD) (Green, 1989; Green et al., 1996) framework with several usability criteria suggested by the International Organization for Standardization (ISO) (ISO 9126-1, 2001; ISO 9126-3, 2003; ISO 9126-4, 2004) as its instrument. Unlike the previous surveys, the instrument was used to guide the discovery of features rather than as a means of assessment.

The following section explains the technical aspects of the survey's preparation and execution. Section 7.2 presents the results of the survey. Section 7.3 discusses the data analysis and contribution of the survey. Section 7.4 explains several threats to the validity of the results. Finally, Section 7.5 concludes the chapter with a summary of the main findings and future work.

7.1. Objectives and Methodology

7.1.1. Motivation

The objective of the survey was to capture some experience of using verification tools that accompany methods such as UML-B. It was not the intention of the survey to investigate every possible instance of verification tools and delineate their strengths and weaknesses. While the tools undergo improvements over time, new tools are also introduced. Any extensive investigation on the tools is seen as not worthwhile as they could become obsolete and overwritten by others. Rather, the survey aimed to identify basic features that should be present in verification tools for them to become usable. The survey started the investigation with two instances of verification tools, namely ProB and B-Toolkit. As a study on two instances could not reveal all features, the findings from the investigation are left open for further investigation and discussion in future where they can be validated and expanded.

Despite being individual tools, ProB and B-Toolkit are indeed related. B-Toolkit verifies a model by generating proof obligations, which can be discharged through automatic and interactive provers (B-Core, 1999). The automatic prover discharges the proof obligations automatically. On the other hand, the interactive prover requires user intervention for the proof activities to complete, which can be complex and time consuming. ProB uses model checking technique (Clarke et al, 1999) for exploring exhaustively the finite behaviour of a model, an animator for executing the operations and a graphical tool for displaying the reachable states and transitions. As verification tasks in B-Toolkit can be difficult to perform, ProB aims to eliminate some non-trivial errors before more complicated interactive proof is attempted in B-Toolkit (Leuschel et al., 2003). Due to this reason, the tools were considered together as one object of study in the survey.

The survey was qualitative in nature where its analysis was mainly interpretive. Based on the captured user experience, the analysis aimed to identify a set of features that are believed to be important for ensuring the usability of verification tools. The survey concerned the usability assessment from the perspective of new users. New users in this context refer to developers who are new to not only verification tools but also model verification tasks. To achieve this objective, the survey employed the following research question.

What are the important features/functionality that should be available in verifications tools for them to be usable (i.e. understandable, learnable, operable and attractive) to new users?

In one sense, this survey also aimed to extend the findings of the survey on UML-B described in **Chapter 6**. One of the theories generated from that survey is as follows:

Theory 2: The integration of semi-formal and formal notations requires strong support from the environment

(a) Support from the internal devices that accompany the method, which should be functionally effective and efficient
(b) Support from the external devices such as documentation and training, which should be comprehensive and useful
(c) Supporting tools should be easy-to-learn and easy-to-use to facilitate understanding and application

As a theory, the *strong support from the environment* is described in a general way in order for it to be applicable to as many instances as possible. In order to understand the phenomenon better, this survey attempted to address some aspects of sub-theory (*a*) and (*c*) above. It aimed to explore the kind of internal support required to facilitate the understanding and application of the tools and their underlying methods. Supporting tools for UML-B are two-tier, that is, before and after the transformation to a formal model. The support necessary before the transformation has been described in **Chapter 6**. In this chapter, the investigation focuses on the support provided by the verification tools after the transformation.

7.1.2. Materials and Approach

Similar to the previous surveys, the survey instrument was based on the Cognitive Dimensions of Notations (CD) usability framework. The framework comprises fourteen dimensions, which acted as the variables in the survey. In addition, several usability criteria of ISO were also included. The survey adopted the grounded theory approach for the data analysis (Strauss et al., 1998). CD and the justification of approach selection have been explained in the earlier chapters. In this chapter, the attention is given on the analysis and interpretation of the data.

The questions for the survey were constructed by following the proposed CD questionnaire (Blackwell et al., 2000). The proposed CD questionnaire was tailored and modified slightly to reflect the characteristics of verification tools. The survey used the CD framework, albeit concerns tool environment more than notation. This is because verification tools such as ProB and B-Toolkit are designed to support activities concerning models that describe system functionality. The tools interact actively with the notations used in the models to ensure they specify the system functionality accurately and consistently. Therefore, it would be awkward to investigate the tools solely without considering the notations that they interact with.

There were nineteen questions in the survey. Fourteen questions reflected the fourteen dimensions of the CD framework, four questions represented the ISO usability criteria and one question gathered suggestions for improvement. The questions were presented in random order without following a specific sequence. Unlike the previous surveys, the answers were entirely subjective as the survey aimed to capture the features and their impacts rather than to gauge the degree of CD and ISO satisfaction. The details of the questions used in the survey can be found in the **Appendix E**.

Prior to survey questionnaire distribution, the validity and accuracy of the questions were reviewed by a focus group. There were four people involved in the process, who would use the results of the survey. The purpose of the review was to identify any missing and unnecessary questions as well as ambiguous questions and instructions.

7.1.3. Participation

Sixty-three out of one hundred potential participants responded to the survey. The response rate was therefore sixty-three percents. They were Undergraduate and Masters students of Computer Science and Software Engineering courses at the University of Southampton, who registered for the “Critical System” course in Spring 2006 and 2007 (ECS, 2007). Master students constituted one-third of the participation. Four of the participants out of ten potential participants were

Undergraduate students of Computer Science course at the University of Surrey (UoSurrey, 2007). The participants from the University of Southampton responded to the questionnaire on ProB whereas the participants from the University of Surrey responded to the questionnaire on B-Toolkit. The international students, who came from outside the United Kingdom constituted half of the participation. The proportion of women to men was 1:4.

The survey questionnaires were distributed to those potential participants because they were independent users of ProB and B-Toolkit, who used the tools for the first time for model verification tasks. The participants had some practical experience of using the tools when participating in the survey. Specifically, they used the tools to animate and verify the models that they developed during the course. In Southampton, the participants used ProB while the participants from Surrey used B-Toolkit. The participants had gone through courses on formal methods at some points of their studies. The participation was voluntary where the questionnaires were completed anonymously and submitted at the end of the respective course. The participants were aware that the survey was intended for research purposes.

The survey adhered to the University's ethical policies and guidance for conducting research involving human participants (UoS, 2007). In particular, the materials and procedure used in the survey had been reviewed and approved by the University's Ethics Committee. A *Participant Information Sheet* and a *Consent Form* were enclosed together with the questionnaire during the survey distribution. The participants were advised to read and understand the information contained in the *Participant Information Sheet* before deciding to participate. To participate, the participants were asked to sign the *Consent Form* and submit it together with the completed questionnaire to the course leaders.

The participants were in the final semester of the respective course and thus had reasonable amount of experience and knowledge of software development. Some of the Masters students had some work experience for at least one year. They may have not been exposed to all possible experience in industry. As far as the

objective of the survey is concerned, they represented closely software developers who are new to verification tasks and tools.

7.2. Results

The survey questions assessed the functionality of verification tools, namely ProB and B-Toolkit. The following sub-sections present the responses received from the respondents for each of the questions in the survey questionnaire. The first fourteen sub-sections reflect the dimensions of the CD framework while the subsequent four sub-sections represent several usability criteria suggested by the ISO. The last sub-section is comments for further improvement.

The data presented in the tables are the summary of the responses received from sixty-three respondents. To avoid redundancy, similar responses are conceptualised and grouped using appropriate descriptions. As the proportion of respondents between B-Toolkit and ProB is 1:15, more data are available and presented under the *ProB* column than the *B-Toolkit* column in the tables. As mentioned earlier in the **Motivation** sub-section, ProB and B-Toolkit are two verification tools that complement each other. Moreover, the survey aimed to capture as many features as possible. Therefore, the data between the *ProB* and *B-Toolkit* columns should not be compared as to determine which tool is better. Rather, they should be considered together as a set of data that best describe the phenomenon.

7.2.1. Visibility and Juxtaposibility

The question **(1a)** of the survey assessed the ability of the tools to allow the user to view and search their various features when working with a B model. The Table 7.1 below shows the summary of responses.

ProB	Btoolkit
PROS	
• Simple interface and headings; clear layout • Menu is organised based on task ordering (i.e. *Animate* before *Verify*) • Use colour-coding (e.g. keywords are presented in colours) • Splitting the screen into panes (i.e. text editor and animation – *StateProperties*, *EnabledOperations* and *History*)	• Quite simple layout • Self-explanatory headings • Multiple panes that represent different views • Use colour-coding (e.g. origin of constructs) • Online help is available whenever needed
CONS	
• Features and menu are not so intuitive for beginners (i.e. users need to know what to do and where to find) • Interface is not quite standard like normal applications such as *Windows*; no toolbar • Some features are difficult to configure (e.g. *Preferences*) and not obvious (e.g. *User mode*) • Panes are not flexible (i.e. cannot be closed and resized) • Some features use external applications (i.e. *Animate* uses *dotty* graph) – need to be configured correctly • Limited shortcuts keys and not obvious; no right click • Very limited online help	• Interface is not quite standard like normal applications such as *Windows* • Features look simple but the scope is actually much larger – need some time to explore various menus and options • Environment configuration has to be correct for features to work properly

TABLE 7.1: Answers for the "Visibility" dimension

The question **(1b)** of the survey assessed the ability of the tools to allow the user to view several parts of a B model or several B models at the same time for comparing purposes. The Table 7.2 below shows the summary of responses.

ProB	BToolkit
PROS	
• Easy to navigate between the text editor and animation panes (i.e. *StateProperties*, *EnabledOperations* and *History*) – easy viewing of important areas and debugging • *Recent File* eases the navigation between different models	• Easy to navigate between panes • Panes are resizable
CONS	
• Difficult to view different parts of the model at once – have to launch another application or copy to another text editor and print • Cannot display more than one model at the same time – have to close and reopen models, launch another application or copy to another text editor and print • Layout is convenient but the panes cannot be resized; cannot collapse/hide unused panes • Text editor is too simple; limited text editor pane allow only a small part of the model can be viewed at one time; need scrolling quite a lot • No "Find and Replace" to find specific parts; operation names are not highlighted for easy identification	• Difficult to view different parts of the model and different models at once - have to copy to another text editor or print • Not enough screen space to display multiple panes and pop-up windows

TABLE 7.2: Answers for the "Juxtaposibility" dimension

7.2.2. Viscosity

The question **(2)** of the survey assessed the degree of effort required by the user to perform a change in a B model by using the tools. The question required the respondents to indicate how easy to make the changes and why. The Table 7.3 below shows the summary of responses.

ProB	BToolkit
PROS	
• Use colour-coding helps identifying parts to change • Error messages are displayed whenever the model is reopened • *Animate* and *Verify* are useful for testing the model after changes are made • *Summary of B syntax* is available	• Most editing tasks involve using the text editor pane • Clicking buttons for verification • The effect of a change on the entire system is demonstrated before the decision is made to commit the change (i.e. *Status*)
CONS	
• Manual (i.e. typing word by word); syntax is very rigid; no real time syntax checking; need to trace the whole model whenever a change is made to know how the change affect other parts • Verification is time consuming • No "Find and Replace", "Go To", "Undo", "Redo", "Cut" and "Paste" causes editing difficult – have to use external text editor such as *Notepad* and *Word* • *Save and Reopen* is easy but it is a bit strange to reopen the model in order to verify it; *Save* feature is useless; no "Save As" • Cannot display two models at the same time; limited text editor pane space • Unclear and unhelpful error messages; do not point to where the error is in the model • No refactoring feature • Pop-up window for information has to be closed while editing (i.e. *Summary of B syntax*)	• Syntax is very complex and character sensitive • Verification task is difficult and time consuming • Once a model is introduced, all modification should be done in the tool due to its configuration management - otherwise, may cause problems • Many pop-up windows have to handle during some operations • Error messages sometimes can be difficult to understand • Use of some buttons in pop-up windows (e.g. <OK> and <CANCEL>) are misleading/unusual

TABLE 7.3: Answers for the "Viscosity" dimension

7.2.3. Diffuseness

The question **(3)** of the survey assessed whether the tools let the user did what he/she wanted to a B model reasonably straightforward. The question required the respondents to indicate the actions that took them more time and effort to accomplish. The Table 7.4 below shows the summary of responses.

ProB	BToolkit
• Animate the model manually to find where the invariant violations occur as *Random Animation* feature does not provide any information • Ensure preconditions satisfy invariant, run and check all possible scenarios; require too much prior knowledge; too much details need to be examined at one time • Sort out errors	• Get used to symbolic commands (e.g. *anm*, *sts*, *ppf* etc.) to execute things • Understand error messages • Discharge proof obligations

TABLE 7.4: Answers for the "Diffuseness" dimension

7.2.4. Hard Mental Operations

The question **(4)** of the survey assessed the degree of mental processes required for the user to animate, verify and analyse a B model using the tools. The question required the respondents to indicate how difficult to comprehend what was happening during the tasks and why. The Table 7.5 below shows the summary of responses.

ProB	Btoolkit
PROS	
• Features are easy to locate from the menu and self-explanatory • Easy to understand if know B well enough • Different methods to verify and animate that can be used together • Features such as *Temporal Model Check* and *Random Animation* are simple and executed automatically (e.g. find deadlocks, invariant violations and inspect existing nodes) • *Analyse Invariant* helps users to understand why invariants are violated; *Current State* is useful for viewing operations with valid preconditions • Graphical viewing is useful to view what is happening • Easy to view operations in animation panes (i.e. *StateProperties*, *EnabledOperations* and *History*); *Backtrack* is useful for identifying invariant violations	• Compiler-style output is helpful • Animation (i.e. *anm*) helps to understand what is happening • *AutoProver* and *BToolProver* are useful for simple proof obligations
CONS	
• Error messages are difficult to understand and not helpful; not clear and do not make sense most of the time; lengthy and complicated list of errors • No indication of what is happening and why (i.e. only indication that errors have been found, invariants have been violated and how many states have been visited); animation is not performed systematically (i.e. no order) • Difficult to view a large model in graphs; cannot compare graphs (e.g. *View Visited States* and *Reduced Visited States*) – need to zoom-in and zoom-out constantly and scrolling • Difficult to keep track large model – need to trace a long list of operations and can be confusing • Animation panes (i.e. *StateProperties*, *EnabledOperations* and *History*) are too textual – should be made graphical	• *InterProver* is actually quite useful but it is hard to execute • Error messages are difficult to understand instantly – need some time to become familiar • Backtracking the output can be confusing • Better to have a window that shows all system states at any given point • Quite hard to understand how to uncover inconsistencies • Type errors take some time to spot • Performance of provers can be inefficient due to bugs and incomplete proof rule library

TABLE 7.5: Answers for the "Hard Mental Operations" dimension

7.2.5. Role Expressiveness

The question **(5)** of the survey assessed how easy the user could recognise and interpret the roles of available features in the tools. The question required the respondents to indicate features that were particularly difficult to interpret and why. The Table 7.6 below shows the summary of responses.

ProB	BToolkit
• Roles of some features are not clear (e.g. *Advanced Find*, *Refinement Check*, *Constraint based Checking*, *Show Typing*, *Analyse Assertions*, *Show Filtered Conjuncts*, *Compute Coverage*, *Find a Non Resetable Node* , *Debug* and *Subgraph of Nodes Satisfying GOAL*) and complex to understand how to configure (e.g. *Preference*) • Roles of animation panes (i.e. *StateProperties*, *EnabledOperations* and *History*) is not obvious initially – should have "Tool Text Tip" to briefly explain their purposes • Roles of colours in graphs seems to be misleading (i.e. red normally indicates errors but it means non-deterministic nodes in the tool) • Error messages are not clear • Existence of "Cut" and "Paste" are not obvious	• Roles are fairly easy to interpret if users know what to do • Some error messages are not descriptive enough to successfully discharge proof obligations • Only obvious and common features are frequently used

TABLE 7.6: Answers for the "Role Expressiveness" dimension

7.2.6. Provisionality

The question **(6)** of the survey assessed the flexibility of the tools. The question required the respondents to indicate how well the tools allowed them to play around with a B model without being sure what the effect would be. The respondents were required to state which parts of the tools that allowed or prevented them to do so. The Table 7.7 below shows the summary of responses.

ProB	BToolkit
PROS	
• Error messages displayed each time the model is reloaded • Animation panes (i.e. *StateProperties*, *EnabledOperations* and *History*), *Random Animation*, *Temporal Model Check* and *Analyse Invariant* features are useful for testing ideas • Can comment out sections that are not needed; can set preferences for animation • Different methods to animate and verify	• Online help is available on the *Help* menu; brief description of each utility is available upon clicking • The current number of outstanding proof obligations and total number of proof obligations are displayed for each construct
CONS	
• Error messages are unclear and not descriptive enough; not pointing to the position • Text editor is too simple; no "Undo" and 'Save As" prevent experimenting; limited screen size • *Save and Reopen* the model each time to find the errors • Very limited online help	• *InterProver* is difficult to understand and operate – have to know what rules to add to discharge the proof obligations • Proving is time consuming and can be daunting • Multiple directories of machines is not supported

TABLE 7.7: Answers for the "Provisionality" dimension

7.2.7. Error Proneness

The question **(7)** of the survey assessed the tendency of the tools to induce mistakes. The question required the respondents to indicate mistakes that were particularly common or easy to make. The Table 7.8 below shows the summary of responses.

ProB	BToolkit
• Comments with ";" is not acceptable • Missing and excess punctuation (e.g. ";", "&", "/\", "||"); missing brackets or "IF..THEN..END" • Mapping and relations (i.e. domain, range) • B syntax maps to ASCII	• B syntax problems • Confusion between specification and code

TABLE 7.8: Answers for the "Error Proneness" dimension

7.2.8. Progressive Evaluation

The question **(8)** of the survey assessed the ability of the tools to allow the user to evaluate his/her work in progress at any time. The question required the respondents to indicate whether or not it was possible to stop modelling at any

time to check their work so far. The respondents had to state why if it was not possible. The Table 7.9 below shows the summary of responses.

ProB	Btoolkit
• Incomplete model may not be animated and verified (depending on what information is missing) and produces error messages • Model with only a few operations may be animated but it would be very limited and some operations may not be reachable • Can animate and verify only if there are no syntax and type errors • It is not always clear why the syntax is incorrect	• Proving is manageable only if the model contains enough information • Proof library may not contain certain rules to discharge proof obligations – need to introduce rules

TABLE 7.9: Answers for the "Progressive Evaluation" dimension

7.2.9. Premature Commitment

The question **(9)** of the survey assessed whether the tools allowed the user to go about any task in any order or enforced the user to make decisions prior to modelling. The question required the respondents to indicate the ordering and decisions that they needed to make in advance, if any. The Table 7.10 below shows the summary of responses.

ProB	BToolkit
• Cannot add operation until the necessary sets, types, variables, constants, initialisation, invariants are decided and declared • Can write any order but to animate and verify, the model must contain necessary information • Animation does not work if verification is failed • Have to fix the syntax errors before the model can be verified and animated • Have to *Save and Reopen* each time changes are made • Some features (e.g. *Find non deterministic nodes*) and graphs display make only sense after the model has been animated and verified	• B model structure • Machines come first before refinement and implementation • *Commit* then *Analyse* then *Animate* • Discharge obligations in *AutoProver* then *BToolProver* then *InterProver* • Exhaustive environment configuration and setting – need to set up configuration based on the environment used for things to work later

TABLE 7.10: Answers for the "Premature Commitment" dimension

7.2.10. Closeness of Mapping

The question **(10)** of the survey assessed the closeness of mapping between features in the tools and B modelling. The question required the respondents to indicate whether they found any features in the tools that seemed to be a strange way of working with a B model. The Table 7.11 below shows the summary of responses.

ProB	BToolkit
• *StateProperties* pane displays set elements separately rather than a list • No obvious way to check syntax; No mechanism for checking the post conditions satisfy invariants and guards • No support for *While* loop	Maximum length for the name of a construct is 20 characters

TABLE 7.11: Answers for the "Closeness of Mapping" dimension

7.2.11. Hidden Dependencies

The question **(11)** of the survey assessed whether there was any relationship between two parts such that one of them was dependent on the other but the dependency was not fully visible. The question required the respondents to indicate whether or not they found any feature dependencies in the tools. If they did, the respondents had to state how visible the dependencies and what parts that were involved. The Table 7.12 below shows the summary of responses.

ProB	Btoolkit
• *Analyse* features (e.g. *Analyse Invariant* and *Analyse Properties*) can be executed and useful only after executing *Animate* feature • Graphs display can be executed only after animation	Should exit a session by selecting *Utils/Exit* – otherwise, the development will be locked and data may be damaged

TABLE 7.12: Answers for the "Hidden Dependencies" dimension

7.2.12. Secondary Notation

The question **(12)** of the survey assessed the ability of the tools to allow the user to provide supporting information to a B model by using any medium other than the B syntax. The question required the respondents to indicate whether or not they could make notes or convey extra information beyond the model to themselves. The respondents had to state the possible actions, if any. The Table 7.13 below shows the summary of responses.

ProB	BToolkit
• Can change fonts and colours of the syntax, but setting them is a bit tricky (i.e. *Preferences* feature) • Use comments (i.e. /* */)	Use comments

TABLE 7.13: Answers for the "Secondary Notation" dimension

7.2.13. Abstraction Gradient

The question **(13)** of the survey assessed whether the tools enforced any level of grouping mechanism. The question required the respondents to indicate whether the tools insisted they start the task by defining or grouping things before they could do anything else. The Table 7.14 below shows the summary of responses.

ProB	BToolkit
• Sets, variables, invariant, initialisation etc. have to be grouped together – the template is useful • The model must have "MACHINE" or "REFINEMENT" or "IMPLEMENTATION" heading at the beginning and "END" at the end	• Correct B model format and syntax are insisted • Initialisation must establish the invariant and each operation re-establish the invariant • Static parts

TABLE 7.14: Answers for the "Abstraction Gradient" dimension

7.2.14. Consistency

The question **(14)** of the survey assessed whether similar functionality in the tools were presented in a similar manner. The question required the respondents to indicate whether or not they found any features in the tools that were similar in

functionality but the tools made them appear different. The respondents had to state what the features were, if any. The Table 7.15 below shows the summary of responses.

ProB	BToolkit
Random Animation and *Random Animation (10)* features	NIL

TABLE 7.15: Answers for the "Consistency" dimension

7.2.15. Tool Accessibility

The question **(15)** of the survey assessed whether it was easy to become familiar with the tools and to be able to use it in their task efficiently without referring to the documentation. The respondents had to state why. The Table 7.16 below shows the summary of responses.

ProB	Btoolkit
• Interface, menu structure and features are quite intuitive and simple • Error messages are not descriptive	• Clear menu structure and headings • Proving takes some time to master

TABLE 7.16: Answers for tool accessibility

7.2.16. Tool Usefulness

The question **(16)** of the survey assessed the usefulness of the tools. The question required the respondents to indicate whether the tools helped them to find problems in a B model. The Table 7.17 below shows the summary of responses.

ProB	BToolkit
• Animation panes (i.e. *StateProperties*, *EnabledOperations* and *History*), *Animate* and *Verify* and *Analyse* features are useful • Error messages are very complicated and lengthy unnecessarily – difficult to understand and confusing • Error messages are not descriptive; too vague/general – no explanation of the meaning of the messages • Error messages indicate what the tool is expecting but do not indicate what the problem is – need to go through the model • Identifies errors but not location • Error messages are displayed at one; individual errors are not separated clearly – difficult to read • Error messages are not quite reliable/correct and misleading sometimes • Online help for correcting errors are not available	• Error messages show where the problem is (i.e. what line) – good • Provers contain bugs and thus error messages can be complicated unnecessarily

TABLE 7.17: Answers for tool usefulness

7.2.17. Usefulness of Documentation

The question **(17)** of the survey assessed the usefulness of the tools' available manual and documentation. The Table 7.18 below shows the summary of responses.

ProB	BToolkit
• *Summary of B syntax* is very useful but it is quite lengthy and cannot be viewed while editing model; no ASCII versus B reference • Online help is not helpful; not enough information about the tool and its features and how to use them • Only few examples available • Need more help on B notation – should provide useful links for information on B	• Reference on syntax, extensive and cross-linked documentation are useful • Demonstrations are available • Click on utilities may display brief description • Lengthy and quite difficult to understand

TABLE 7.18: Answers for usefulness of documentation

7.2.18. Tool Purposefulness

The question **(18)** of the survey assessed whether the tools helped the user to understand and learn the B method. The Table 7.19 below shows the summary of responses.

ProB	BToolkit
• Animation panes (i.e. *StateProperties*, *EnabledOperations* and *History*), *Animate* and *Verify* and *Analyse* features are useful for understanding model • Allows experimentation to know what works and does not work - better than on paper; learn through practice • Error messages are not helpful and misleading (i.e. without telling how to solve them and why they occur, a syntax error can produce lines of error messages) • Mapping between B syntax and ASCII symbols can be confusing	• Learn through practice • Need more examples to be more helpful

TABLE 7.19: Answers for tool purposefulness

7.2.19. Further Improvement

The question **(19)** of the survey required the respondents to indicate features that they used most often. The question also provided the respondents an opportunity to suggest any possible improvement that could be made on the tools. The Table 7.20 below shows the summary of responses.

ProB	BToolkit
Features often used: • Animation (i.e. *Random Animation*; *View Visited Space*) • Verify (i.e. *Temporal Model Check*; *Find a Non-deterministic Node*; *Find a Non-resetable Node*) • Analyse (i.e. *Analyse Invariant*; *Compute Coverage*) • Animation panes (i.e. *StateProperties*, *EnabledOperations* and *History*) • *Save and Reopen* feature **Improvement:** • Allow saving graphs; improved graphs display • Allow printing • Descriptive and specific error messages – what, where, how; break down errors • Better editor features (i.e. "Undo", "Copy", "Paste", "Find and Replace", "Save As" etc.; automatic indentation) • Layout in Animation panes – should use "tree with branches" • Wider editor space, resizable and split windows for easy viewing and switching; hide and view panes • Improved interface (i.e. toolbar with "Tool Text Tip"; Java-like GUI) • Shortcut keys and right-click • Real time debugger – no need to save and reopen model; real time syntax highlighting and missing character checking • Can open more than one model at the same time; having tabs • Useful documentation • Installation should be easier; no command prompt running at the back • Accept any symbols for comments • Include FAQ and discussion/mailing list; reference on B	**Features often used:** • Animator (i.e. *anm*) • Analyser (i.e. *anl*) • Provers (i.e. *AutoProver*; *InterProver*; *BToolProver*) **Improvement:** • Improved interface with font colours • More understandable provers • Windows that display all states during animation • Can run on *Windows* platform

TABLE 7.20: Answers for further improvement

7.3. Analysis and Discussion

The survey aimed to identify the important features or functionality that should be available in verification tools for them to be usable to new users. To achieve this objective, the survey employed the grounded theory approach for the data analysis. The approach enables the categorisation of features based on specific properties and dimensions. Unlike the previous surveys, the use of CD and ISO's usability criteria was not intended to be the properties that determine the categorisation. Rather, they were used as a means for the analysis to identify common features that emerged from the data. In other words, they acted as a medium for a broad-brush analysis. The captured features may not be necessarily sufficient. However, they are believed to be the essential conditions for verification tools to be usable. The assumption behind the analysis is that the frequently emerged features are indeed the ones that are highly valued and expected by users from such tools.

Based on the responses summarised in the previous section, a set of feature properties have been identified. The properties enable a formation of several discrete categories. The properties are indeed interrelated, thus the categories are connected through them. Each property has dimensions that describe its specific usability characteristics. There are three main categories discovered during the analysis, namely *Interface*, *Work Utilities* and *Resources Management*. *Work Utilities* is the main functionality of verification tools. To perform as intended, *Work Utilities* requires the interaction of *Interface* and *Resources Management*. The properties of *Work Utilities* are therefore interrelated with the properties of the other two categories. The following sub-sections list the categories and properties. The corresponding interrelated properties are stated in the parentheses in the table of **Category 2 (C2): Work Utilities**.

7.3.1. Category 1 (C1): Interface

This category refers to the structure and organisation of screen layout and utilities. The Table 7.21 below lists the necessary properties and dimensions.

Menu concerns the presentation and arrangement of utilities so that they can be easily searched and interpreted. Utilities should be defined and grouped in a logical way with simple and self-explanatory headings. The tasks involved in verification tools are normally complex, thus only the necessary utilities should be presented. As formal modelling imposes specific rules and sequence of events, it may be better if the utilities are arranged and controlled in certain orders. This is to ensure that users are clear of what to be done without being overwhelmed with superfluous utilities. To expedite tasks, commonly used utilities should be made available in mediums other than the menu bar. Moreover, the utilities must represent closely the principles of the underlying methods so that users can smoothly apply the methods. As formal modelling is mainly rigid, users should be offered with supporting utilities that can be set as needed. This is to ease understanding of the models.

Panes should be made flexible enough for users to view different parts of a model and switch between different models. This is particularly essential when performing model editing and modification. Formal models such as B are lengthy, thus the tools should facilitate the viewing of distant parts of a model. In fact, B involves several stages of development that represent different perspectives. Users are more likely to compare the model of one stage to the other. While it is necessary to be able to view several parts or models at the same time, the tools should also allow users to resize, open and close the panes as needed. This is to avoid cluttering the screen with many views.

It is a norm for tools to communicate with users when certain operations are executed. *Dialogues* are intended to inform users about the current and future actions and to display information for reference. To be useful, the dialogues should be available only when they are expected. They also should behave conventionally such as in other tool environments to avoid confusion and

unnecessary mistakes. For instance in many environments, <OK> button always mean users agree with the action or acknowledge the situation. Hence, it would be awkward for it to represent otherwise. Dialogue windows normally require users to select one of the options or buttons before proceeding with the next action. However, some dialogue windows contain information that guides users through the process. These windows should be allowed to remain while users executing the action. This is particularly necessary for formal modelling due to the complexity of the tasks.

Property	Dimension
Menu	• Utilities are defined and grouped using clear and self-explanatory headings • Available utilities can be easily searched and inferred from the headings • No superfluous and redundant utilities • Utilities are arranged and controlled by task ordering (i.e. enabled/disabled based on task at hand) • Commonly used utilities are available as icons on toolbar, shortcut keys and right-click options • Utilities match closely the principles of underlying method • Supporting utilities are available and can be set (i.e. add notes/comments, preferences for editing/viewing models etc.)
Panes	• Width of panes can be resized • Panes can be closed/collapsed/minimised and reopened/expanded/maximised • Allow "Split View" to view different parts of the same model • Allow "Cascade/Tile" or tabs to view different models • Allow scrolling
Dialogue	• Appear appropriately as in standard practice (i.e. to inform status, to confirm decision, to display information) • Use conventional buttons with standard meanings (i.e. <OK> to confirm and <CANCEL> to defer etc.) • Not all dialogues should be closed to proceed (e.g. windows contain information such as online help can be displayed and remain while model editing)

TABLE 7.21: Properties and dimensions of "Interface"

7.3.2 Category 2 (C2): Work Utilities

This category refers to the utilities required for formal modelling. The Table 7.22 below lists the necessary properties and dimensions.

The notation used in formal models is normally textual. Thus, it is essential for users to be able to do *Editing and Formatting* to the text. The tools are generally expected to perform similar operations such as in other text editors or word processing applications. At the very least, the appearance of the text can be changed, its location can be moved and searched, and users can revert to previous actions. Users also should be able to treat models as document files where they can be changed to different forms and locations. To facilitate the editing and formatting task, the most common utilities should be handy. Moreover, the tools should provide enough working space for performing the task and facilities for users to communicate informally the model to themselves. Reference should be available whenever needed.

Being able to check the accuracy and consistency of a model is the main advantage of formal modelling. Formal notations are very rigid and specific. There is always a tendency for users to use symbols incorrectly, specify inappropriate data types and overlook keywords. Thus, it is essential for the tools to perform *Syntax Checking/Analysis* automatically with the necessary explanation of what have been found. Users must be informed appropriately about any misuse and missing elements. The checking acts as the first error filter before more complex tasks are performed. Reference should be available whenever needed.

Verification tools should have an *Animation* facility, which allows users to visualise model behaviour under the stated conditions and rules. The facility may be available in several different mediums and can be done automatically and semi-automatically. Automatic animation is only feasible for accurate and consistent models. Therefore, semi-automatic animation is useful for users to identify specific points where rules violation and unintended behaviours occur. Backtracking should also be available for the purpose. As troubleshooting can be complex, the tools should have a mechanism to guide users through the process.

To ease understanding, the animation should use graphical representation with appropriate colour coding. Models can be large, thus the facility should facilitate the viewing. Users should be informed about any errors encountered, current status and possible effects. Reference should be available whenever needed.

Verification is regarded as the most difficult task to perform on a formal model. It is where the accuracy and consistency of the model are confirmed. Therefore, the tools should be able to prove the model automatically as much as possible. Otherwise, users should be guided so that they can better understand their own model and the verification process. Understanding is crucial, as some aspects of the task cannot be performed automatically. For instance, an incomplete model cannot be verified, thus users must be aware of the missing elements. Users should also know how to glue the new elements to the ones that are already specified in the model so that their conditions and actions do not conflict with each other. *Animation* can also ease the understanding through model visualisation. Several different approaches may be available for users to verify the model. Visual indicators such as colours or objects can be used to indicate important elements. Elements involved in the verification task should be visible and the task is performed as efficient as possible. Similar to *Animation*, users should be informed about any errors encountered, current status and possible effects. Reference should be available whenever needed.

Some formal methods are invented to support several stages of development cycle. For instance, B encourages its abstract models to be refined. A refined model at a sufficiently low level can be translated automatically into code. Verification tools that support such methods should thus faciliate *Code Generation*. Ideally, users should be provided with several options of implementation. At the very least, the tools should include the implementation language that supports the method best. Similar to other tasks, users should be informed about any errors encountered, current status and possible effects. Reference should be available whenever needed.

Property	Dimension
Editing and Formatting	• Text can be formatted (i.e. size, colour etc.), edited (i.e. cut, paste, undo, redo etc.) and searched (i.e. find and replace, go to etc.) (**C1: Menu**) • Model/file can be organised and manipulated (i.e. save as different file, print etc.) (**C1: Menu; C3: File Management**) • Commonly used utilities for formatting and editing are available on the toolbar as well as shortcut keys and right-click options (**C1: Menu**) • Pane for editing is wide for viewing most parts of a model (**C1: Panes**) • Informal information can be added to model and editing preference can be set (**C1: Menu**) • Reference is available whenever needed (**C3: Online Documentation**)
Syntax Checking/Analysis	• Syntax are checked automatically and instantly (e.g. missing brackets and punctuation, typing errors on keywords, incorrect types etc.) with explanation of what have been found (**C1: Dialogue; C3: Error Management**) • Unresolved syntax and type errors are communicated clearly and specifically (**C1: Dialogue; C3: Error Management**) • Performed before animation and verification (**C1: Menu**) • Reference is available whenever needed (**C3: Online Documentation**)
Animation	• Automatic and semi-automatic with information of what happening; Semi-automatic animation is guided (**C1: Dialogue; C3: Error Management**) • Different approaches to animation are available to view animation from several perspectives (**C1: Menu**) • Use graphical representation with appropriate colour coding to demonstrate animated elements (**C1: Menu; C3: Interoperability**) • Animated elements can be viewed easily (i.e. zooming, side-by-side etc.) and manipulated (i.e. print, save) (**C1: Panes; Menu; C3: File Management**) • Encountered errors are communicated clearly and specifically (**C1: Dialogue; C3: Error Management**) • Current status and possible effects are communicated (**C1: Dialogue**) • Backtracking is possible but guided with explanation (**C1: Dialogue**) • Reference is available whenever needed (**C3: Online Documentation**)
Verification	• Automatic and semi-automatic with information of what happening; Semi-automatic verification is guided (**C1: Dialogue; C3: Error Management**) • Different approaches to verification are available to verify model from several perspectives (**C1: Menu**) • Use appropriate colour coding or objects to indicate and highlight elements/process (**C1: Menu**) • Verified elements can be viewed easily (**C1: Panes**) • Perform within reasonable time (**C3: Interoperability**) • Encountered errors are communicated clearly and specifically (**C1: Dialogue; C3: Error Management**) • Current status and possible effects are communicated (**C1: Dialogue**) • Reference is available whenever needed (**C3: Online Documentation**)
Code Generation	• Model may be transformed to code automatically (**C3: Interoperability**) • Different types of code generation are available (**C1: Menu**) • Encountered errors are communicated clearly and specifically (**C1: Dialogue; C3: Error Management**) • Current status and possible effects are communicated (**C1: Dialogue**) • Reference is available whenever needed (**C3: Online Documentation**)

TABLE 7.22: Properties and dimensions of "Work Utilities"

7.3.3. Category 3 (C3): Resources Management

This category refers to the management of entities that are related to the execution of work utilities. The Table 7.23 below lists the necessary properties and dimensions.

Users should be given several options of running the tools. The tools should cater several different *Platform*s so that users can select the one that suits their environment. The installation and configuration should be made as simple as possible and should be supported by comprehensive documentation.

Formal models normally evolve from one stage of development to the other where the latter stage depends on the former. Therefore, it is necessary for the tools to have a *File Management* mechanism to manage and monitor the gradual development. Furthermore, any changes made in one stage should be reflected in other related stages to ensure model consistency. Users should be informed of the process and have the opportunity to decide.

Some utilities may need the services provided by other independent applications. For instance, the animation facility may need visualisation software. *Interoperability* should be ensured by seamlessly integrating separate applications as one unit. Moreover, internal and external utilities should be made compatible with each other to ensure process efficiency. If the independent applications have to be obtained by users themselves, the information about the location of the resources should be made available. The documentation of how to install and integrate the applications with the tools should also be provided.

Error Management is of critical importance in verification tools. Formal methods in general are difficult to grasp instantly where users' rate of learning can be slow. The tools should generate error messages that do not only explain explicitly what goes wrong but also facilitate learning. To avoid unnecessary mental burden, the error messages should be made simple, precise and timely. Some errors have to be solved by users themselves due to incomplete specification of requirements. Even so, users should be provided with guided error messages to help identifying

missing information. Other than those errors, the tools should be able to solve. To be effective, the tools must include a proof library that contains as many rules as possible so that it can detect most inconsistencies and inaccuracies.

The complexity of the tasks requires *Online Documentation* to be easily accessible to users. The documentation should not only cover the functionality of the tools but also the underlying methods and how the tools support the methods.

Property	Dimension
Platform	• Tool can be set up in various platforms • Installation and configuration can be easily executed and supported by comprehensive documentation
File Management	• Files are managed and monitored systematically • Consistency among interrelated files are ensured • Changes are controlled, checked and reported
Interoperability	• External applications are integrated seamlessly and operate as intended • Different elements (internal and external) interact with each other in an efficient manner • Installation and configuration can be easily executed and supported by comprehensive documentation
Error Management	• Error messages are descriptive: what errors, which parts, why they occur and possible solutions • Error messages are simple but precise • Error messages are displayed at the right time and place • Error messages are displayed clearly so that they are legible • Almost complete and reliable proof library is available for performing tasks and generating reliable/correct error messages
Online Documentation	• Simple and comprehensive documentation on the available utilities • Summary of syntax used in model and its mapping with keyboard entries (e.g. B syntax and ASCII, and special symbols) • Some external links about information on method (e.g. hypertext links to B method and tools), discussion forum or "Frequently Asked Questions" • Some examples and demonstrations about the tool and method • "Tool text tip" or brief description are available for utilities on the toolbar and elements on any other bars • A shortcut key to online help is available • Reference on correcting common errors

TABLE 7.23: Properties and dimensions of "Resources Management"

The categories, interrelated properties and dimensions described above are intended to act as a guideline for designing verification tools. As the survey was the first attempt to understand the usability of such tools, the guideline is not expected to be comprehensive and complete. In fact, it considers only the most important features, which are believed to particularly influence the usability of the tools. However, the captured features have been found to be similar with a survey conducted on visualisation tools (Kienle et al., 2007), which verification tools should also be included as they support the visualisation of elements.

To improve the accuracy of the guideline, further investigation and discussion are needed so that it can be confirmed and refined. In fact, the guideline is presented in an abstract way in order to embrace all possible verification tools. It is assumed that any design plan of a particular verification tool should elaborate the dimensions more specifically to fit the tool's context of use. Some trade-offs are expected where certain dimensions may need to be compromised in order to gain the benefits of others. For instance, online documentation and error messages may need to be lengthy in order to be comprehensive. They may thus become difficult to view on screen. Similarly, in order to view several elements at the same time, the screen space has to be divided into several panes. Tool designers therefore have to decide the best compromise.

7.4. Threats to Validity

The four kinds of validity that must be protected in empirical studies are discussed below (Cook et al., 1979a; Perry et al., 2000). Most of the threats are similar in studies presented in earlier chapters.

7.4.1. Internal Validity

This validity concerns whether there is a clear cause and effect relationship between the dependent and independent variables (Gauch, 2000). It determines whether the usability assessment of the verification tools could be influenced by any other factors than the tools themselves.

7.4.1.1. Instrument

The survey aimed to discover as many features as possible that can ensure the usability of verification tools. It used the CD framework and several usability criteria of ISO as the survey instrument, which may have not been sufficient to explore all features. On the other hand, it is better to start with some criteria that could guide the investigation and act as a discussion tool. At the very least, they allow some aspects to be discovered which can be further explored in future.

7.4.1.2. Selection of Respondents

Some of the respondents were students from the university where the research was conducted. Therefore, their answers might have been bias either in positive or negative ways. They however were independent users, who had no personal interest with the technologies involved or direct contact with the research. To reduce the threat, the subjects were advised to give opinions and comments as sincerely as possible.

The participation from Southampton responded to ProB while Surrey responded to B-Toolkit questionnaire. Bias may have been introduced by “who use what”. It

might be that the results could be different if other arrangements were used. For instance, reverse assignment was employed or respondents responded to both ProB and B-Toolkit questionnaire. Due to time and resource constraints however, the arrangement was seen as appropriate.

7.4.1.3. Sample Size and Response Rate

Sixty-three out of one hundred potential participants responded to the survey. The response rate was therefore sixty-three percents. Although the participation received for ProB was quite high, the participation for B-Toolkit was rather small. Only four participants responded to the survey. The data may not be representative enough to describe the usability of B-Toolkit. However, that was the best participation that the research could get for two consecutive Spring terms.

Brief identity screening was done on the students from Southampton who did not respond. No particular pattern was identified. It was not possible to identify students from Surrey due to the university's data protection.

7.4.1.4. Diffusion or Imitation of Response

The respondents were in contact with each other. There was a risk that they shared or influenced each other's comments, which could not be controlled. During the analysis, two cases where two questionnaires had identical answers were found. Only one of two similar questionnaires was considered for the analysis. As the survey concerned qualitative data not quantitative, the impact was seen as minimal.

7.4.2. External Validity

This validity refers to the ability to generalise the results and conclusions of a study to other populations and conditions than those used in the study (Gauch, 2000; Yin, 2003).

7.4.2.1. Students as Respondents

The respondents of this survey were students. They may have not represented software developers as they were less experience and perhaps were likely less motivated. However, the respondents were in the final semester of their course and had reasonable amount of experience and knowledge of software development. Despite being students, the respondents were considered as the most appropriate candidates for the survey because they were new users of ProB and B-Toolkit and verification tasks. Hence, they fitted the objective of the survey.

The results could have been better if software professionals were included. From the researcher's own experience however, it was hard to get new and independent B tools users. Most B tools users are experienced and involved in research involving B. Thus, they could introduce bias.

7.4.2.2. Toy Problem

The coursework that required the respondents to use the tools was not large and may have not been representative of real software systems. This includes both universities: University of Southampton and University of Surrey. However, the coursework was believed to be sufficient for the respondents to experience the tools and verification tasks.

7.4.2.3. Selection of Instances

The survey considered only two instances of verification tools. In fact, they are tools of one particular formal method, namely B. The results therefore may be bias and not represent all possible verification tools. Because of this reason, the findings of this survey are considered as tentative where they should be confirmed and refined in future studies.

7.4.3. Construct Validity

This validity concerns the establishment of correct operational measures for the concepts being studied (Yin, 2003). It is necessary to ensure that the dependent variables are valid measures and the measurement process is conducted appropriately.

7.4.3.1. Dependent Variables

The dependent variables of survey were the fourteen dimensions of CD and four usability criteria of ISO. These variables were seen as appropriate for measuring the usability because they covered both notational and operational aspects. Their validity and appropriateness as a measure of usability has been assessed to some degree by their authors.

7.4.3.2. Analysis Process

One person did the data interpretation and analysis. Although this may pose a threat however, it can be regarded as negligible as the person was an independent user.

7.4.3.3. Nature of Study

Surveys and qualitative measures by their nature are retrospective. Therefore, there was a risk that the respondents responded based on what they thought they did rather than what they actually did. Advising the respondents to complete the survey questionnaire as soon as they did the modelling task could have reduced this threat, as the respondents still remembered of what he or she found during the task.

7.4.4. Conclusion Validity

This validity concerns the ability to draw the correct conclusion about relations between the object of study and the outcome of the survey.

7.4.4.1. Heterogeneity of Respondents

The respondents might have different ability and experience. Thus, there was a risk that the results might have been affected by individual differences. This could not be avoided. As a qualitative study, the variation however could provide richer data for the analysis.

7.4.4.2. Familiarity of Respondents

The respondents were given a period of one term to explore and use the tools. Even so, they were not expected to use the tools everyday. The results might have been different if the respondents were given more time. The aim of the survey was to capture the experience of using the tools from new users' perspective. Therefore, the allocated time frame was seen as adequate and realistic for the purpose. Beyond the time frame, the respondents could be regarded as intermediate users rather than beginners.

7.5. Conclusions and Future Work

This chapter has presented a survey conducted on ProB and B-Toolkit. They represented two instances of verification tools. The survey attempted to understand the nature of experience of using verification tools. It aimed to explore basic features that are expected to be present in verification tools for them to be usable to new users. The survey used the Cognitive Dimensions of Notations (CD) framework and the International Organization for Standardization's (ISO) usability criteria as the medium of exploration. The use of the grounded theory approach for the data analysis enables the identification of abstract concepts and properties of usable verification tools. The concepts and properties form a guideline, which can be used by tool designers when designing verification tools.

There are three main elements that could potentially affect the usability of verification tools. They are the interface that organises the tools' utilities, the

tools' main utilities and the management of resources that support the main utilities. Each of the elements has specific properties and dimensions for it to be usable. The elements however are interrelated through their properties. For example, for a property to achieve a desired dimension, it needs the support from at least one of the other properties. Moreover, one dimension may need to be compromised in order to achieve other dimensions. The three elements therefore should be considered together when designing verification tools. Tool designers should aim for dimensions that best suit their tools' context of use.

The survey proposes a design guideline for verification tools to be useful. As the guideline was generated based on two instances of verification tools, it may not cover all the necessary usability features. Therefore, future studies are encouraged to investigate other verification tools so that the guideline could be refined and extended. This includes verification tools of other formal methods such as Z. The validity of the guideline could also be improved by extending the survey to experienced software developers who are new to verification tools and tasks. Meanwhile, studies could also extend the guideline by considering the design aspects more technically. For example, the input and output devices, and dialogue techniques that best present the utilities could be investigated. Such studies require theories and principles from Human Computer Interaction (HCI) discipline.

Chapter 8

Evaluation and Recommendation

Previous chapters have presented the empirical assessments conducted in this research. The assessments explored the efficacy of a software development method that combines the use of semi-formal and formal notations, namely UML-B (Snook et al., 2006). The assessments involved controlled experiments and surveys conducted on UML-B's model and modelling environment respectively. In this chapter, the findings of the assessments are gathered to guide general evaluation of the strengths, weaknesses, threats and opportunities of UML-B. The evaluation is intended to generate usability theories of methods such as UML-B. It is believed that the aspects discussed in this chapter are applicable to other methods that have motivation and characteristics similar to UML-B. In particular, methods that incorporate a semi-formal notation into a formal notation to produce a model that is not only precise and consistent but also accessible.

In the following Section 8.1 to 8.6, the evaluation of UML-B is organised into *What*, *Why*, *Who*, *Where*, *When* and *How* aspects. Finally, Section 8.7 concludes the chapter with a summary of the main points.

8.1. What

"Integrated methods" is the term used in the research to mean development methods that combine the use of semi-formal and formal notations in creating software artefacts. The artefacts originate as early as the system specification stage. Semi-formal notations are representations that provide a set of graphical symbols to represent specific system elements. Some examples of semi-formal notations include Entity-Relationship Diagram (ERD) (Chen, 1976; Elmasri et al., 2001), Data-Flow Diagram (DFD)(Yourdon, 1989), Unified Modelling Language (UML) (OMG, 2006) and Open Modelling Language (OML) (Firesmith et al., 1998). On the other hand, formal notations such as Z (Spivey, 1992, Bowen, 1996) and B (Abrial, 1996) use textual mathematical symbols and interpretation to describe a system. The notations have rules for determining the grammatical well-formedness of sentences, rules for interpreting sentences in a precise and meaningful way, and rules for inferring useful information from a specification (van Lamsweerde, 2000). Besides the format, formal notations are essentially different from semi-formal notations because the former define properly the intended properties and behaviour of system elements.

Theoretically, "Integrated methods" encompass any possible combination of semi-formal and formal notations. For the integration to be useful however, there are some compatibility criteria that the participating notations have to meet. First, even if not entirely, there are some basic principles in both notations that fit together. For instance, if the semi-formal notation is best suited for data flow representation, the formal notation should support the characterisation of data entities and transitions. Similarly, if the semi-formal notation supports object-oriented development, the formal notation should possess certain features of objects. In the case of UML-B for example, B has many features similar to UML such as encapsulation of operations with associated state variables and identification of entities with specific relationships. Second, the integration must be complementary where the notations appear as one joined representation rather than two different ways of presenting the same information element. Each notation highlights different types of information that it suits best. The graphical

symbols of the semi-formal notation can be used to illustrate elements that are readily interpreted such as grouping of entities, relationships between entities and transitions between states. The formal notation is used to describe literally the properties, behaviours and constraints of the entities that cannot be illustrated by the graphical symbols. The integration of the two qualitatively different notations is enabled through specific rules that define the complementary roles of each notation.

The primary objective of "Integrated methods" is to promote the use of formal or mathematically-based methods in software development. While being able to produce a reliable system (Craigen et al., 1995; Hinchey, 2002), barriers towards formal methods are recognised (Finney et al., 1996a; Finney, 1996b). The aim of "Integrated methods" is to enable formal methods to become more accessible. UML-B for instance, is intended to overcome barriers towards B so that it can be widely used in industry (Snook et al., 2006). Therefore, the approach taken by "Integrated methods" such as UML-B differs from the ordinary view of graphical and textual representations integration such as UML and its formal syntax; Object Constraint Language (OCL) (Warmer et al., 2003). OCL is designed as an annotation notation for UML diagrams so that the diagrams can be more expressive and precise. The intention is primarily for the presentation and interpretation of information where OCL (formal notation) augments UML diagrams (semi-formal notation). In contrast, "Integration methods" go beyond just that. The aim is to produce a formal model that is accurate and precise through thorough validation and verification activity. The activity however is easier to accomplish only if users understand their own models well. The use of semi-formal notation helps in achieving the aim by improving a formal model's accessibility among its users.

"Integrated methods" are geared towards facilitating verification activity, which is generally assisted by special tools. Ideally, such tools should be able to discharge the inaccuracies and inconsistencies of a model automatically. As the tools perform the verification activity based on predefined rules, hypotheses and theorems of proving, there are limitations on the way they can prove a model. To enable the tools to discharge errors as efficiently as possible, "Integrated methods"

manipulate the participating notations into forms that are easier for proving. Due to this reason, there are differences in styles when the notations are used together and when they are used individually. This particularly affects the semi-formal notation as the verification tools are designed to interpret the formal notation. For instance, to appropriately represent the mapping of sets in B that can easily be verified, the use of class multiplication and association in UML-B is quite different from the conventional UML. This claim is supported by the results of the surveys (**Chapter 4** and **6**) conducted in this research.

The first step in using a formal method is to employ its formal notation in specifications. Therefore, the role of "integrated methods" begins from the specification and design stage. The sources or inputs for conducting specification and design tasks using the methods are user requirements of a system. The intermediate products of the methods are formal specifications or models, which act as the means of communication and reference among stakeholders. If the formal method is employed extensively, the specifications can generate executable code and test cases for later stages.

8.2. Why

"Integrated methods" aims at promoting the individual strengths of semi-formal and formal notations and complementing one notation's weaknesses by the strengths of the other. Formal notations have the ability to increase a model's precision and consistency (Plat et al., 1992; NASA, 1998), as special-purpose tools can systematically verify them using specific proof theory. The notations however are generally difficult to comprehend because of their mathematical symbols and rules of interpretation (Carew et al, 2005). Semi-formal notations employ graphical symbols. The graphical symbols, whose concepts are similar to objects in the real world, are mainly intuitive (Bauer et al., 1993; Stenning et al., 1995). Semi-formal notations however are not as expressive as formal notations as graphical symbols cannot illustrate system properties completely (Petre, 1995).

Moreover, graphical symbols cannot be verified systematically to confirm the accuracy and consistency of a model.

"Integrated methods" are capable of improving formal or mathematically-based software development. In particular, the methods promote formal models' comprehensibility while yet offering greater confidence that correct models are developed. The methods could ensure an effective validation task for formal models by getting stakeholders more involved in the process. Stakeholders' involvement during the validation task is critical for the successful software development (Standish, 1998; 2001). The visualisation aspect of "Integrated methods" provided by the semi-formal notation could encourage the participation of stakeholders as it improves the readability of formal models. With reasonable hours of formal training, namely less than ten hours, stakeholders should be able to interpret the models. The readability of the models could also facilitate efficient maintenance where maintainers can easily grasp the required information prior to maintenance tasks. These claims are particularly supported by the results of the controlled experiments (**Chapter 3** and **5**) conducted in this research.

8.3. Who

"Integrated methods" are appropriate for users who favour a natural approach of formal modelling. Specifically, the methods mainly benefit new formal method users who are already accustomed to using visual modelling techniques. To these users, the use of graphical symbols together with textual mathematical symbols at abstraction level is regarded as more approachable than using the mathematical symbols entirely. This claim is supported by the results of the surveys (**Chapter 4** and **6**) conducted in this research.

Combining semi-formal and formal notations in "Integrated methods" allow the strengths of both notations to be reinforced. But more importantly, it permits one notation's limitations to be compensated by the other. Such methods need users to be aware of the strengths and weaknesses of the participating notations. The

awareness enables users to understand the motivation behind the integration, appreciate the effort and form reasonable expectation.

The integration of semi-formal and formal notations in "Integrated methods" involves the use of two individual notations, which are qualitatively different in the way they present information. There are significant differences on how textual and graphical representations illustrate system elements, as mentioned earlier. Moreover, the individual notations are normally accompanied by certain methodology or rules that guide their application. For instance, B method defines the ways B notation should be employed in specifications to enable verification and refinements in later stages. In contrast, UML encourages system development through visualisation using different perspectives. "Integrated methods" combine not only the individual characteristics of the notations but also their specific strategies of supporting system development. Users of such methods therefore should have the capacity to grasp the underlying principles and methodology of both participating notations.

"Integrated methods" entail some adjustments to be made on the participating notations. For instance in UML-B, a set of appropriate B/Event-B syntax is attached to UML diagrams for specifying textual constraints and actions. Despite being a mathematically-like notation, the syntax has object-oriented elements to suit UML's style of modelling. The UML diagrams are matched to the B/Event-B syntax through specific rules of integration. The integration rules cause the notations to operate differently together from how they would individually. Rather than thinking of the notations independently, users of "Integrated methods" thus need to understand how both notations work together under the rules. As each notation represents one aspect of a system that is not represented by the other notation, the roles that each notation plays in the integration should also be understood.

The models of "Integrated methods" are likely to contain several elements of information as one representation illustrates one aspect of a system. This requires the models to have scattered parts that should be related and interpreted together to form a meaning. For example, a UML-B model has UML-like diagrams and

B/Event-B syntax. In fact currently, there are several types of diagrams in UML-B that present system information from several different perspectives. The information presented in one diagram should be related with the one presented in the other diagrams. Users with high spatial ability, who are able to conceptualise the spatial relations between different types of representation and parts, are believed to be at an advantage when dealing with such models.

Successful technology transfer requires not only a new idea but also a receptive audience (Pfleeger et al., 2000). From the marketing perspective, the audience can be divided into two categories, namely *Early Market* and *Main-stream Market* (Moore, 1991). The *Early market* audience requires little evidence of the efficacy of a new technology. They therefore comprise people who are eager to try any new technology other than they use normally. They also include people who are driven by their perception of a new technology's potential benefits, which is build from others' success stories. This audience is willing to take great leaps and welcomes major changes to the current practices. In contrast, the *Main-stream Market* audience requires more evidence. They are more cautious, careful and sceptical people who will only adopt a new technology if it is necessary. They prefer to make minor changes where the new technology enhances rather than replaces the current practices. In the context of UML-B and "Integrated methods", they are more likely to be accepted by the *Early Market* audience. This is because the technology and idea are very new, which require more evidence to be gathered. Besides, there are a number of requirements that need to be met for adopting the methods, which will be explained later.

8.4. Where

"Integrated methods" combine the use of semi-formal and formal notations in a model, which is later transformed to a formal model that can be verified and executed. As this involves interaction between different representations and transition between several stages, manual operation of such a process can be

complicated and troublesome. "Integrated methods" therefore should be highly automated and equipped with strong supporting tools.

Supporting tools are of critical importance. The tools encompass the internal devices that support the modelling activity and the external devices that prepare users with the necessary knowledge and expertise. The internal devices include the editor in which the model is developed, a translator that transforms the model to a formal model completely and verification tools that prove the accuracy and consistency of the formal model. The external devices include comprehensive documentation about the methods' principles and the features supported by the tools, and user training. The tools should assist users to capture the essence of the methods and use them correctly. Complexity of dealing with multiple notations and modelling elements should be reduced through a simple and user-friendly modelling environment. The tools are also expected to operate as efficiently as possible. These claims are supported by the results of the surveys (**Chapter 4** and **6**) conducted in this research. Some features that are necessary for the tools to become usable have also been discovered (**Chapter 7**).

8.5. When

"Integrated methods" are indeed formal or mathematically-based methods. Although "Integrated methods" use graphical symbols together with textual mathematical symbols, they gradually produce formal (mathematical) models. Formal methods in general emphasise the need of having reliable systems. Systems developed using formal methods are believed to be safe and predictable. This is because the specified system properties and behaviours are proven systematically to be accurate and consistent under all known conditions.

Reliable systems assured by formal methods come with a price. Formal development can be exhaustive and resource extensive. System properties and behaviours have to be defined unambiguously and cross-checked across several levels of abstractions and refinements. The process is interactive where system

elements are added, removed and manipulated during verification activity based on predefined hypotheses. Although verification tools should be able to generate hypotheses and prove them automatically, there are still occasions where users have to intervene. This is particularly true when system requirements are incomplete or contradict one and another. Users should be aware of various possible conditions and understand how the system behaves under those conditions. This is to enable them to formulate hypotheses for the verification tools to base on.

Formal development is difficult for users to master instantly. The verification task particularly is time consuming and troublesome as users need to understand not only the system to be developed but also how to apply the methods correctly. Formal notation in itself is not easy to grasp. Users need to understand the meaning and properties of the notation and how it may be manipulated. Users thus need strong support from the environment. It has been suggested that users should have regular access to experts during formal development to ensure the successful use of the methods (Hinchey, 2002; Bowen et al., 1995; 2006). This means organisations may need to allocate a budget to hire experts for projects using formal methods. Effective and efficient tools that support the development should also be available and usable. Moreover, users should be prepared with the necessary knowledge prior to using the methods. Some time has to be allocated for users to acquire expertise and fluency to express new problems, solutions and proofs using the methods. These requirements have been discovered from the surveys (**Chapter 4**, **6** and **7**) conducted in this research.

The development process using "Integrated methods" can be different from other process models. Such methods emphasise the need of having an accurate and consistent specification of the system through thorough validation and verification activity. The assumption is that whenever the front-end process has been carefully handled, the end product can be assured to be reliable. The formality imposed at early stages can highlight issues that might not be discovered at later stages. Therefore, the early development stages using the methods can be longer where several levels of activities have to be executed before the artefact can move to the next stage. In fact, if the front-end process is executed properly as intended, the

later stages such as coding and testing can be shortened. In the case of UML-B for example, the generated and verified B/Event-B model can be translated automatically into executable code if it has been refined at a sufficiently low level. While intergration testing may still be required, the code may not need extensive unit testing as its internal accuracy has been systematically verified.

The above arguments indicate that "Integrated methods" do require high operational cost and investments. Similar to conventional formal methods, "Integrated methods" are believed to be particularly suitable for the development of critical systems where system safety and reliability is the main priority. Such systems cannot afford to fail. The necessity to have safe-guaranteed systems that reliably operate undermine the high cost of its development. Organisations that opt into using "Integrated methods" should be willing to make some adjustments to their current process model to suit the methods' way of operating. The allocation and distribution of resources may also need rearrangement. Organisations are advised to have a cost estimate and some idea of anticipated costs before embarking on such methods.

8.6. How

System development using "Integrated methods" consists of several phases. System properties and behaviours are firstly illustrated using semi-formal (graphical symbols) and formal (mathematical) notations. Both notations are used together to create a model of the system based on predefined rules of integration. In this first phase, the produced model is suitable for stakeholders such as clients who need to only interpret the model for user requirements validation purposes.

One objective of "Integrated methods" is to assure the accuracy and consistency of the system to be developed. To achieve that, validation and verification activity takes place as early as the specification stage. Despite being formal, the initial model produced by the methods contains graphical symbols that might not be verified systematically using tools. The methods thus have a mechanism to

transform the graphical formal model to a textual formal model that is viable for verification. This is achieved by having a translator that transforms each graphical symbol to the equivalent textual formal representation. The transformation is based on the mappings defined in the methods' principles and rules of integration.

Automated tools support verification activity in "Integrated methods". The tools interpret the model and verify its accuracy based on predefined hypotheses and proving theorems available in the proof library. Users are informed of any detected syntax errors and semantics that cannot be proven. Users should make the corrections on the model and introduce new hypotheses to the proof library for the tools to discharge the errors. Verification tools are generally sensitive and use specific techniques to discharge errors efficiently. Users should be accustomed to the techniques. The tools could assist users in understanding the model and verification task by having an animation facility. The facility provides users with some kind of visualisation that shows the behaviours of the model under the specified conditions. One of the surveys conducted in this research (**Chapter 7**) has discovered a set of basic features that are necessary for the tools to become usable. The model that has been verified by the tools is considered as accurate and consistent, which may be translated to code.

In general, there are two technical aspects that are important in "Integrated methods", namely environment and notation. The role of environment has been discussed in earlier paragraphs. Notation is vital as it is the starting point of the whole idea of integration. Most software systems are large and complex. No single notation will address all aspects of a complex system. The participating semi-formal and formal notations should be suitable for the integration where they complement each other to fit the system that they are meant to describe.

The suitability of the notation used in "Integrated methods" can be evaluated through its expressiveness and effectiveness (MacKinlay, 1986). Expressiveness refers to the ability of the notation to represent the intended information whereas effectiveness concerns the efficacy of the notation as a means of representing information. Notation expressiveness and effectiveness can be assessed empirically based on the tasks that the notation uses. As the notation is used to

describe a problem domain in a model, two major tasks include model interpretation and model creation (Gemino et al., 2004). Model interpretation involves understanding of the domain being represented by interpreting the model. Model creation involves communicating one's understanding of the domain by presenting it in a model. This research for instance assessed the notation used in UML-B empirically for both model interpretation and model creation tasks. The controlled experiments (**Chapter 3** and **5**) assessed the comprehensibility of the notation used in UML-B for validation and maintenance purposes. The surveys (**Chapter 4** and **6**) assessed the usability of the notation and its environment to support the creation of a model.

It has been suggested that the expressiveness of a notation should be considered from a theoretical perspective (Gemino et al., 2003). The theoretical consideration can be used to guide empirical investigations and to suggest ways for creating more effective notations. Ontological evaluation (Weber, 2003) is one such assessment. Ontological evaluation concerns whether the notation has been designed to allow users to create a model that provides clear descriptions of the domain being represented. The evaluation is based on ontology, which is a set of beliefs of what might exist and happen in the domain. Four types of ontological evaluation on notation have been proposed (Wand et al., 1993), which include *Construct Deficit*, *Construct Overload*, *Construct Redundancy* and *Construct Excess*. The evaluation can be extensive depending on the scope of the investigation. The Table 8.1 below provides an example of brief ontological evaluation of UML-B. The intention is to show the applicability of ontological evaluation for identifying some limitations of UML-B's notation. A thorough evaluation could be conducted in future.

	Definition	Evaluation on UML-B
Construct Deficit	One or more ontological construct cannot be mapped to the notational construct (i.e. missing)	**Example:** Operations cannot include other operations within the same *Class* diagram; No bi-navigable associations
Construct Overload	A notational construct maps to two or more ontological constructs	**Example:** Objects in UML-B's *Class* diagram can represent not only entities but also types, events and variables
Construct Redundancy	Two or more notational constructs map to a single ontological construct	**Example:** System behaviours and state transitions can be specified in both *Statechart* diagram and in *Class* diagram
Construct Excess	A notational construct does not map to any ontological construct	**Example:** The use of special purpose syntax: ***self*** and **.** (dot)

TABLE 8.1: Four types of ontological evaluation of UML-B

The effectiveness of a notation can be explained theoretically by employing theories from cognitive sciences and psychology. Model interpretation task involves understanding where the presented information is not only captured but also integrated with other knowledge in the memory. Cognitive theories assume that understanding occurs through a very limited working memory and unlimited long-term memory, which is structured into hierarchically ordered automated mental models (Simon, 1974; Miller, 1956; Chandler et al, 1991; Baddeley, 1992; Sweller, 1999). Cognitive Load theory (Chandler et al, 1991; Sweller, 1999) in particular asserts that the amount of mental load placed on working memory is the critical factor in determining the effectiveness of understanding. This is because for any information to be understood, it must be firstly processed through working memory. If the mental load exceeds the limits of working memory, understanding is hindered.

There are two sources of mental or cognitive load, namely intrinsic and extrinsic (Sweller et al., 1994). Intrinsic cognitive load concerns the intellectual complexity of the information whereas extraneous cognitive load refers to how the information is presented. Intrinsic and extraneous cognitive load together contribute to the total cognitive load involved in understanding.

Intrinsic cognitive load is determined by the degree to which information elements presented in a material interact (Sweller, 1993; Sweller et al., 1994). Some information can be processed sequentially without having to relate its individual elements with one and another. The low interactivity of information elements

causes low intrinsic cognitive load in the working memory. In contrast, some information requires its elements to be processed simultaneously rather than sequentially. The information is highly interactive, as the elements involved cannot be separated as autonomous entities. This causes high intrinsic cognitive load in the working memory. Extraneous cognitive load is determined by the method used to present the information. Each method of presentation varies in extraneous cognitive load. One example of extraneous cognitive load is when the information presented in a material is physically separated. This requires users to use cognitive resources to search and match the related information to form a meaning. This causes high extraneous cognitive load in the working memory as the process of searching and matching information interfere with the process of understanding. The phenomenon is called the *Split-attention* effect (Sweller et al., 1990; Chandler et al., 1991; 1992). It has been suggested that understanding could be enhanced if the related information is physically integrated. This could reduce the use of cognitive resources for tasks other than the understanding itself.

Formal (mathematical) notations in general are likely to cause high intrinsic cognitive load. This is because they involve concurrent interactions between syntactical and semantic elements. Each syntax or symbol of a formal notation has a specific meaning. When a set of syntax is used together, the individual meanings and constraints have to be integrated as joined elements that represent a single interpretation. In the case of UML-B, its model contains formal syntax of B notation. Thus, a UML-B can be regarded as having high intrinsic cognitive load similar to its counterparts, B and Event-B models. However, the results of the controlled experiments (**Chapter 3** and **5**) conducted in this research have suggested that a UML-B model is more comprehensible than a B/Event-B model.

Based on the Cognitive Load theory, the total cognitive load in working memory is a function of both intrinsic and extraneous cognitive loads. A UML-B model differs from a B/Event-B model mainly on the method of presentation. Specifically, a UML-B model contains graphical and textual mathematical symbols whereas a B/Event-B model has only the latter. Considering the fact that both models contain similar textual mathematical symbols (high intrinsic cognitive load), it seems that the use of graphical symbols contributes to the

difference in comprehension. The total cognitive load of a UML-B model is lower than its counterparts due to its low extraneous cognitive load. This indicates that the understanding of a high intrinsic cognitive load notation can be improved if it has a low extraneous cognitive load. For instance, the comprehensibility of a formal notation can be better if it is presented in a way that is easy to perceive.

Even though UML-B causes little extraneous cognitive load, it is worth knowing which parts of UML-B that cause it. A UML-B model has physically separated parts, which contain information that has to be mentally integrated. Users have to search and match the separated information to understand the presented problem domain. This particularly happens between different diagrams, diagrams and their textual specifications and between textual specifications of different diagrams. For instance, a UML-B model requires switching between *Class* and *Statechart* diagrams to link system properties with behaviours and to understand the interaction among classes' behaviours. As one screen cannot display all diagrams and textual specifications, the UML-B modelling environment involves multiple screens or panels that contain different information that has to be combined for understanding. This limitation has been discovered from the surveys conducted in this research (**Chapter 4** and **6**).

Based on the theory of limited cognitive processing capacity of human memory, educational psychology research has investigated the role of rote understanding as an alternative pathway to meaningful understanding especially for novices (Hoosain, 1983; Pollock, 2000). Rote understanding is defined as learning discrete elements of information without the knowledge of the connection between separate elements. In contrast, meaningful understanding involves the learning of not only the individual elements but also the interconnection between them. The research suggests that when novices learn complex information, it would be better for them to be firstly exposed to rote understanding. This means the novices are presented with abstract and discrete information before being exposed to details. The rationale of this claim is that complex information has high intrinsic cognitive load, which involves a large degree of interactivity between elements of information. Novices have very limited knowledge about a specific aspect. If novices are introduced into details directly, the working memory is overwhelmed

with various new elements that have to be processed concurrently. Failing to digest any particular elements may jeopardise overall understanding. This also affects the development of mental models in the long-term memory, which is important for future understanding. Experts on the other hand may not experience overloaded working memory. They already possess specific knowledge, which are organised as structured mental models in the long-term memory. The mental models are ready to be triggered whenever needed. Experts are able to understand complex information much easier than novices as the burden of processing the information is mostly transferred to the long-term memory. The working memory load is reduced and it thus has the capacity to allow understanding to develop easily.

A UML-B model is considered as being complex due to the formality imposed by its notation and high interactivity of various information elements. Stakeholders who are involved with UML-B are believed to be mainly novices. Even though the problem domain presented by a UML-B model may not be too alien to stakeholders such as clients or domain experts, the notation and concepts of UML-B might be quite new to them. This is due to the unfamiliarity of software practitioners towards formal methods in general. Similarly, developers who adopt UML-B may not be familiar with the method and problem domain. The idea of rote understanding discussed above might be applicable to UML-B. UML-B adopts a *Top-down* approach where information is presented in a hierarchical order. For example, the top level *Package* diagram provides an overview of the interaction between *Class* and *Context* diagrams. The highest level of a *Class* diagram illustrates only the interactions between classes. To view the behaviours of a class, the *Statechart* diagram contained in the *Class* diagram has to be selected. This seems to indicate that UML-B reinforces rote understanding where the abstract illustration of a system is presented in front and the details are hidden. Only when necessary, the details are displayed. This claim is supported by the empirical assessments conducted in this research. This may provide an explanation of why a UML-B model is better than an Event-B model in promoting meaningful understanding of presented problem domain among new users, as demonstrated in **Chapter 5**. Unlike UML-B, a purely textual formal model such as B and Event-B illustrates system properties and behaviours at once. This may

cause an overloaded working memory, which hinder novices from absorbing new complex information.

Educational psychology research has recognised the potential of dual modality format as an effective way of presenting information (Mayer, 2001). The research is based on the cognitive theory that suggests the working memory is composed of multiple channels, which process different types of information independently with little interference (Paivio, 1986, Penney, 1989; Clark et al., 1991; Baddeley, 1992). The channels include a visual system for dealing with visual images and an auditory system for processing verbal information. The detailed discussion about the theory and its relation to UML-B has been included in **Chapter 5**. It seems that "Integrated methods" such as UML-B are more effective in portraying information than methods that use single notation such as B. One reason is that "Integrated methods" utilises the capacity of the working memory efficiently.

Cognitive and educational psychology theories have offered some explanation of the effectiveness of notation and presentation method used in UML-B and "Integrated methods" in general. The evidence obtained from the empirical assessments performed in this research supports the theories. Dual modality format such as used in "Integrated methods" seems to be beneficial when the information to be presented is complex. Complex information involves high interactive elements. The use of graphical symbols and textual mathematical symbols can be effective in promoting understanding especially among novices. This is particularly true if the notations illustrate qualitatively different elements in a hierarchical order with minimal use of cognitive resources to search and match the related information. These findings and claims may be preliminary, which entail further investigation. However, they act as a starting point for practitioners to understand the efficacy of "Integrated methods" and how it can be improved.

8.7. Conclusions

This chapter has discussed the strengths, weaknesses, threats and opportunities of development methods that combine the use of semi-formal and formal notations. The discussion is based on the findings of the empirical assessments conducted in the research together with some theoretical explanation found in the literature. The discussion focuses on UML-B as the main object of the research. However, the discussion has been generalised to include other methods similar to UML-B, which incorporate a semi-formal notation into a formal notation to produce a precise and accessible model.

The findings of the research indicate that integrated methods such as UML-B have the potential for being an approachable formal (mathematical) development technique. The methods could increase the correctness of the system to be developed not only through systematic verification but also effective validation by stakeholders. The use of graphical and textual mathematical symbols promotes better understanding than the textual symbols alone as it utilises the capacity of human memory. On the other hand, such methods could incur high investment and operational cost. They require organisations to make adjustments to their current process models and practice. Moreover, users should be equipped mentally with knowledge and expertise, and physically with supporting tools. Such methods therefore are mainly appropriate for safety-critical systems, where reliability is more critical than any other concerns.

The integration of semi-formal and formal notations in itself is not easy to accomplish, as the notations are indeed quite different in principles and application. Any attempt to integrate the notations should be started with reasonable aims. It may however evolve over time. Rather than targeting solving various software problems, such an attempt should focus on specific system aspects that the notations suit best. The attempt should also consider the interaction of the notations with their targeted environments such as supporting tools and intended users.

Chapter 9

Conclusions and Future Work

Software systems are becoming a necessity. Many organisations from various industries such as aeronautics and health care are dependent on software to accomplish critical tasks and solve difficult problems. These organisations cannot afford to have low quality software in their systems, as it will affect not only the performance of the process and the quality of the deliverables but also the safety of the people. Therefore, software quality is regarded as an important and necessary factor for the organisations to maintain their competitive strength and to gain credibility. To remain at the forefront, organisations need to acquire high quality software, which software experts are expected to offer and assure consistently.

Although the essence of having high quality software has been realised, software disasters and user dissatisfaction are still prevalent in the industry (Gage et al., 2004). Increasingly large amounts of resources are being allocated to software projects, yet software quality is no better today than it was some time ago (Whittaker et al., 2003). In fact, delivering software on schedule and within the development budget is still a problem. This issue is partly due to the lack of measurement programmes that explore the efficacy of the available software technology used in software development. Software experts seem to be more interested in inventing technology rather than evaluating it. The situation forces practitioners to continue using technology, whose effectiveness remains mostly un-investigated (Solingen et al., 2001).

It is vital to evaluate various software methods and tools to understand how they can help produce high quality software within the specified budget and schedule. Measurement provides an ideal mechanism for evaluating software technology (Basili et al., 1988). Through measurement, the appropriateness of a technology can be assessed in terms of its effectiveness and usability. Conducting empirical evaluations is how measurement can be applied to achieve the objective. The findings of one single investigation are indeed insufficient to explain a phenomenon. Therefore, an empirical evaluation should include several assessments that investigate the phenomenon from different perspectives using different complementary approaches.

This research aims to address the issues of software quality by evaluating the usability of a software development method empirically through measurement. Usability is related to software quality because it establishes the relationships between a software technology and its application domain. In the following section, the aspects covered in the research are summarised. Section 9.2 lists the research's main contributions and Section 9.3 presents the future work ideas. Finally, Section 9.4 concludes the chapter with a summary of the main points.

9.1. Summary of Research

The research investigates a conceptual modelling method that combines the use of a semi-formal notation, Unified Modelling Language (UML) (OMG, 2006) and a formal notation, B/Event-B (Abrial, 1996; Abrial et al., 1998; 2007). The method is called UML-B (Snook et al., 2006). The main objective of the research is to evaluate the usability of the method. Usability in this respect includes the comprehensibility, learnability, operability and attractiveness of the notation used in the method and the method itself in supporting the modelling process (ISO 9126-1, 2001).

The research is mainly empirical in nature. It comprises a series of controlled experiments and surveys. The controlled experiments evaluated the

comprehensibility of the model produced by the method from stakeholders' perspective for software validation and maintenance purposes (**Chapter 3** and **5**). On the other hand, the surveys were intended to evaluate the usability of the method and the supporting tools from developers' perspective for modelling purposes (**Chapter 4**, **6** and **7**).

In addition to the empirical assessments, a theoretical analysis was also conducted on the method to understand *What*, *Why*, *Who*, *When*, *Where* and *How* factors that contribute to its usability (**Chapter 8**). This provides software practitioners with some understanding of the strengths, weaknesses, opportunities and threats of the method. To answer some of those questions, the analysis explored several theories from the Information Science, Cognitive Science and Educational Psychology disciplines. The analysis together with the supporting evidence from the empirical assessments has generated usability theories and design guidelines for methods such as UML-B. The tentative theories and design guidelines could be tested on other similar methods in future.

9.2. Summary of Main Contributions

The original contributions have been divided into primary and secondary contributions.

- **(PRIMARY) Empirical investigation into the comprehensibility of models that integrate semi-formal and formal notations**

  A series of controlled experiments has been conducted from the perspective of stakeholders who interpret the models for validation and maintenance purposes (**Chapter 3** and **5**). The investigation has provided some evidence of such models' accessibility, which could guide the design of future formal methods.

- **(PRIMARY) Empirical investigation into the usability of methods that integrate semi-formal and formal approaches**

  A series of surveys has been conducted (**Chapter 4** and **Chapter 6**) from the perspective of developers who employ the methods for model creation task. The findings have been used to generate tentative usability theories of such methods. A design profile has also been proposed to act as a general guideline for designing usable methods.

- **(PRIMARY) Empirical investigation into the usability of verification tools**

  A set of surveys has been conducted on two instances of verification tools (**Chapter 7**). Some important features for verification tools to become usable have been discovered. This enables the proposal of a tentative design guideline for usable verification tools.

- **(PRIMARY) Theoretical usability evaluation of methods that integrate semi-formal and formal approaches**

  A theoretical evaluation that explains *What*, *Why*, *Who*, *When*, *Where* and *How* the methods can be usable has been conducted (**Chapter 8**), based on the findings of the empirical assessments and several theories from the literature. The evaluation provides practitioners with some understanding of the strengths, weaknesses, opportunities and threats of such methods.

- **(SECONDARY) Multi-method and multi-discipline approaches to empirical software engineering research**

  The research demonstrates the importance of conducting research using multiple methods as a means for acquiring richer understanding of the

phenomenon under study. It also demonstrates the feasibility of conducting software engineering research using approaches and theories from other disciplines such as Social Sciences (**Chapter 4**, **6** and **7**), Clinical Research (**Chapter 3** and **5**), Cognitive Science and Educational Psychology (**Chapter 5**). The research combines both confirmatory (**Chapter 3** and **5**) and explanatory work (**Chapter 4**, **6** and **7**). The research also demonstrates the importance of confirming empirical results through replications (**Chapter 5** and **6**).

9.3. Future Work

The future work with respect to each empirical assessment conducted in this research has been elaborated in the **Conclusions and Future Work** section of the respective chapters. Therefore, they will not be repeated in this section. The following paragraphs discuss the general future work for the research.

9.3.1. Notations

The findings of the research have indicated that the integration of semi-formal (graphical) and formal (textual mathematical) notations is capable of improving the comprehensibility of formal models, as compared to formal notations alone. Several cognitive theories have been adopted to explain the phenomenon. However, there are several aspects that require further investigations as follows:

- **What are the specific characteristics of the notations (graphical and textual) that ease understanding?**

  The understandability of notations is more likely to be influenced by their specific internal characteristics. It may be that the structural properties such as how the notations physically enhance the presentation of system elements in a model. If the notations could illustrate system elements in patterns that make them easy to perceive, understanding would be facilitated. The Gestalt Laws

(Koffka, 1935) for instance have outlined a set of pattern perception rules, which state graphical objects that are similar, close and move together are perceptually grouped, and continuous lines are perceived more readily than contours that rapidly change direction. There are also other related perception rules that indicate objects within an enclosed region of space are perceptually grouped (Palmer, 1992) and objects connected by continuous contours are perceived as related (Palmer et al., 1994). Moreover, object shape, colour and surface texture also play a role in facilitating understanding. For example, studies have suggested that 3D objects are easier to analyse and remember than 2D (Irani et al., 2000), which were based on the theory that humans perceive objects as composed of simple, linked, 3D solid-shape primitives (Biederman, 1987). Furthermore, certain graphical representations bias towards certain problem solutions (Zhang, 1997). Having a graphical representation per se is thus insufficient. The representation must enable the important patterns to be readily perceived. On the other hand, the degree of which the textual representation carries the meaning and the underlying interpretation rules also affect the understanding. This suggests a hypothesis that two models contain two similar integrated notations will not be *informational equivalent* and *computational equivalent* (Larkin et al., 1987) although they both contain graphical and textual representations. Perhaps the integrated notations that incorporate the human pattern perception theories and fit the problem domain would be more helpful for promoting understanding.

- **Which of the two representations (graphical or textual) contributes more to understanding or attract more users' attention?**

  Integrated methods such as UML-B contain graphical and textual representations that complement each other to achieve the integration objectives. Although both are important, it is hypothesised that one representation contributes more to the understanding than the other. Perhaps one representation is better than the other in attracting users' attention where users depend mostly on it for understanding. Moreover, one representation may be more influential than the other in directing how a model is understood.

One way to test these hypotheses is by using eye movement tracking where the pattern of users' attention when reading a model is recorded using an eye tracker such as Tobii (Tobii, 2007). As eye movement provides insight into attention allocation, it is also possible to infer the underlying cognitive processes (Rayner, 1998). The idea of eye movement tracking has been previously used to investigate how programmers read code (Crosby et al., 1990; Romero et al., 2002; Bednarik et al., 2004). It is possible to apply the idea to investigate how stakeholders read models that combine graphical and textual representations. The findings of such studies could be used to formulate strategies for improving the accessibility of the models.

- **Would the combination of graphical and textual representations be more effective for understanding if it incorporates narrations?**

  The Cognitive Theory of Multimedia Learning (Mayer, 2001) has postulated that the understanding of a material could be facilitated if it utilises the dual channels of information processing in human's working memory, namely verbal and visual. While the visual channel mainly processes images that originated from the eyes, the verbal channel processes sounds that derived either from printed words through the eyes or spoken words through the ears. The empirical assessments in this research have investigated the effects of combining printed images and printed words in presenting information. Future studies may need to explore the effectiveness of having spoken words or narrations together with the printed images to present information. For instance, it would be interesting to investigate whether or not by having narrations with diagrams could improve the comprehensibility of a graphical formal model such as UML-B.

9.3.2. Method and Process

The research investigates an instance of integrated methods that combine the use of semi-formal and formal notations. The investigation covered only the development of conceptual models at the specification and design stage. To

promote such methods to practitioners, further investigations are required as follows:

- **Are the methods that combine semi-formal and formal approaches cost-effective?**

  Future studies may need to carry out cost analysis on integrated methods such as UML-B. This includes cost-effectiveness analysis (Levin, 1983) and cost-benefit analysis (Mishan, 1988). The former involves the comparison of alternatives to determine the most efficient way to achieve the benefits while the latter determines whether the benefits outweigh the costs. Integrated methods are indeed formal methods, which mainly promise the benefits of having an accurate system through thorough validation and verification activities. Cost effectiveness analysis thus compares integrated methods and formal methods to determine which methods could achieve the benefits in a more efficient way. The analysis should also consider the additional benefit of integrated methods, which is being more approachable than the traditional formal methods. On the other hand, cost-benefit analysis investigates the relationship between the cost of using integrated methods and the value of the benefits that results from it. This analysis could determine whether or not it is worth using such methods at all.

  Cost-effectiveness and cost-benefit analyses are likely to involve more than just one project. The analyses might investigate a large project or compare several smaller pilot projects that take different approaches to solving the same software problem. They may also entail following up over a long period of time, to look at the long-term impact of using the methods. It is not possible to use a true experimental design to assess the effect, thus quasi-experimental designs (Cook et al., 1979b) and case studies are more likely to be adopted. Cost-effectiveness and cost-benefit analyses can be complex. They require very sophisticated technical skills and training in methodology and principles of Economics. Thus, future studies that investigate this aspect should not be taken lightly.

- **How could organisations adopt such methods with minimal adjustments to the current process model and practices?**

  Integrated methods such as UML-B are considered as formal, which use mathematical-based notations and rules of interpretation. As described in **Chapter 8**, formal methods have their own strategies for developing software. The methods emphasise the need of having iterative refinement and verification processes. The strategies could be different from the common process models such as Waterfall (Royce, 1970) or even the latest approach such as Agile development (Agile, 2007). Managers in general are reluctant to modify extensively the current development process employed in their organisations. This is because the employed process model has become a part of the organisations' culture and therefore, any major changes on it would be difficult. Future studies may need to investigate how such integrated methods could be adopted with minimal adjustments to the current process models employed by organisations. Guidelines could be proposed where managers are provided with a number of approaches and options for effectively including the methods in the current processes and practices.

9.3.3. Empirical Assessment Approaches

The research combines both qualitative and quantitative approaches in its investigation of UML-B. The investigation consisted of confirmatory and explanatory study where the former was mainly quantitative and the latter was qualitative. As stated in **Chapter 2**, a confirmatory study does not necessarily have to be quantitative and an explanatory study can also be quantitative. Although quantitative data are more likely to necessitate statistical analyses and qualitative data are more apt to justify qualitative data analyses, yet both statistical analyses and qualitative data analyses can be used to explore and to confirm phenomena (Onwuegbuzie et al., 2005). Qualitative data can be used to test hypotheses (Onwuegbuzie et al., 2003; Patton, 1990; Tashakkori et al., 1998). Therefore, future studies may need to look into the possibility of adopting qualitative approaches in confirming hypotheses. Similarly, the qualitative data of

an explanatory study could be quantified and analysed using statistical analyses. To date, those kinds of study are not common in empirical software engineering research. If properly conducted, such studies would contribute vastly to the body of knowledge of empirical software engineering.

9.4. Conclusions

The research aims to assess the usability of an instance of integrated methods, which combine the use of semi-formal and formal notations, namely UML-B. The research has achieved the goal by conducting a series of investigations on the method from different perspectives using different empirical research approaches. The findings of the investigations have provided some explanation of the usability of UML-B. Based on the findings, several usability theories and design guidelines for integrated methods such as UML-B have been proposed. The research provides some directions of future work so that better understanding of the phenomenon could be obtained. In addition, the ways of how the methods and the research could be improved has also been discussed.

Appendix A

Controlled Experiment 1

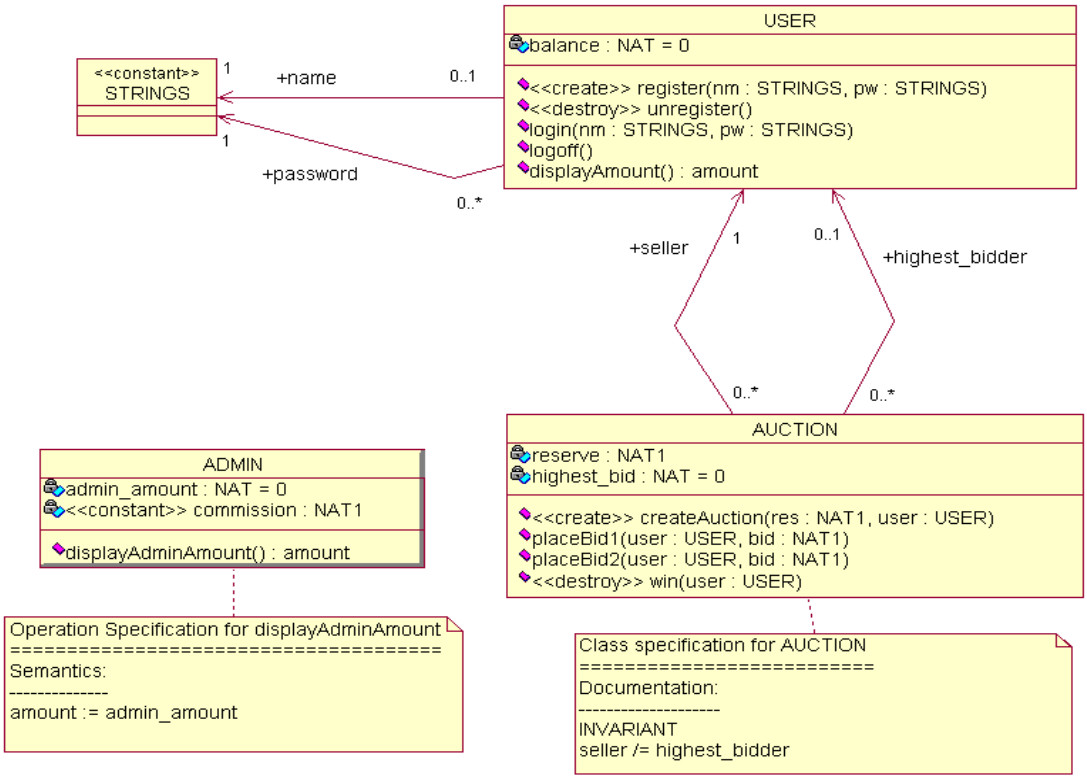
This appendix presents the materials used in the first controlled experiment (**Chapter 3**). This includes the models and questions used in the experiment.

A.1. Models

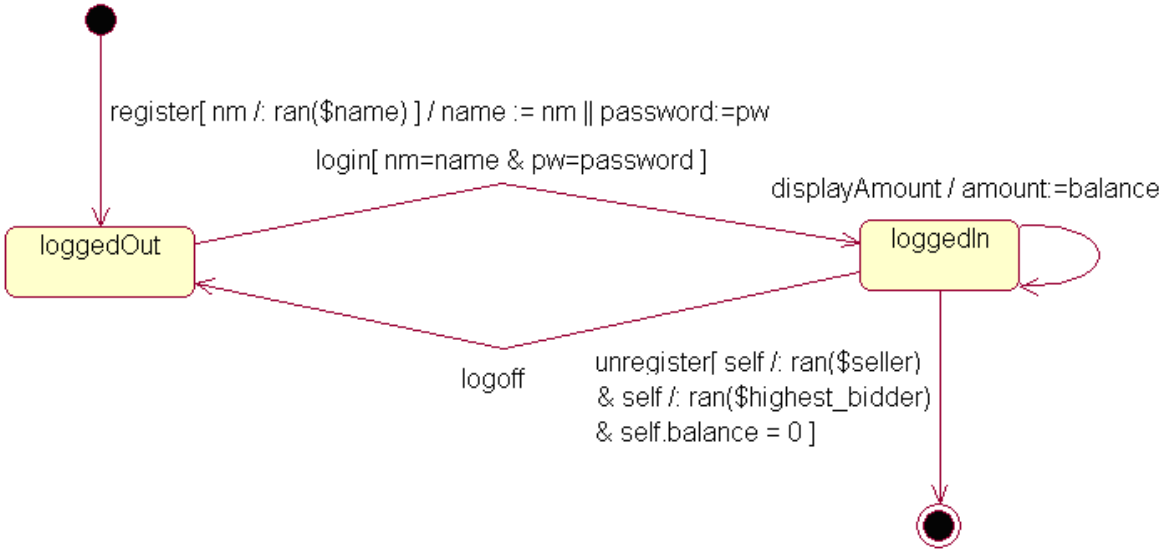
A.1.1. UML-B Models

Case 1: Auction System

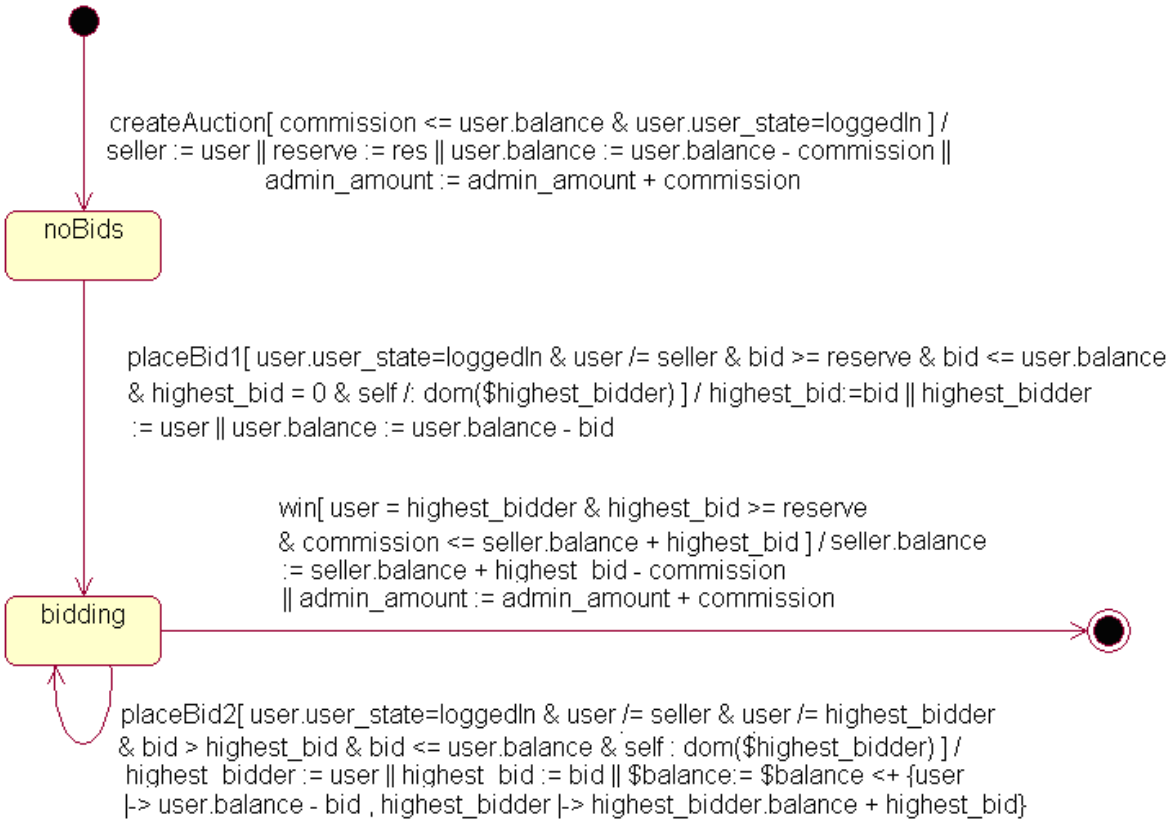
Class Diagram



Statechart Diagram for USER Class

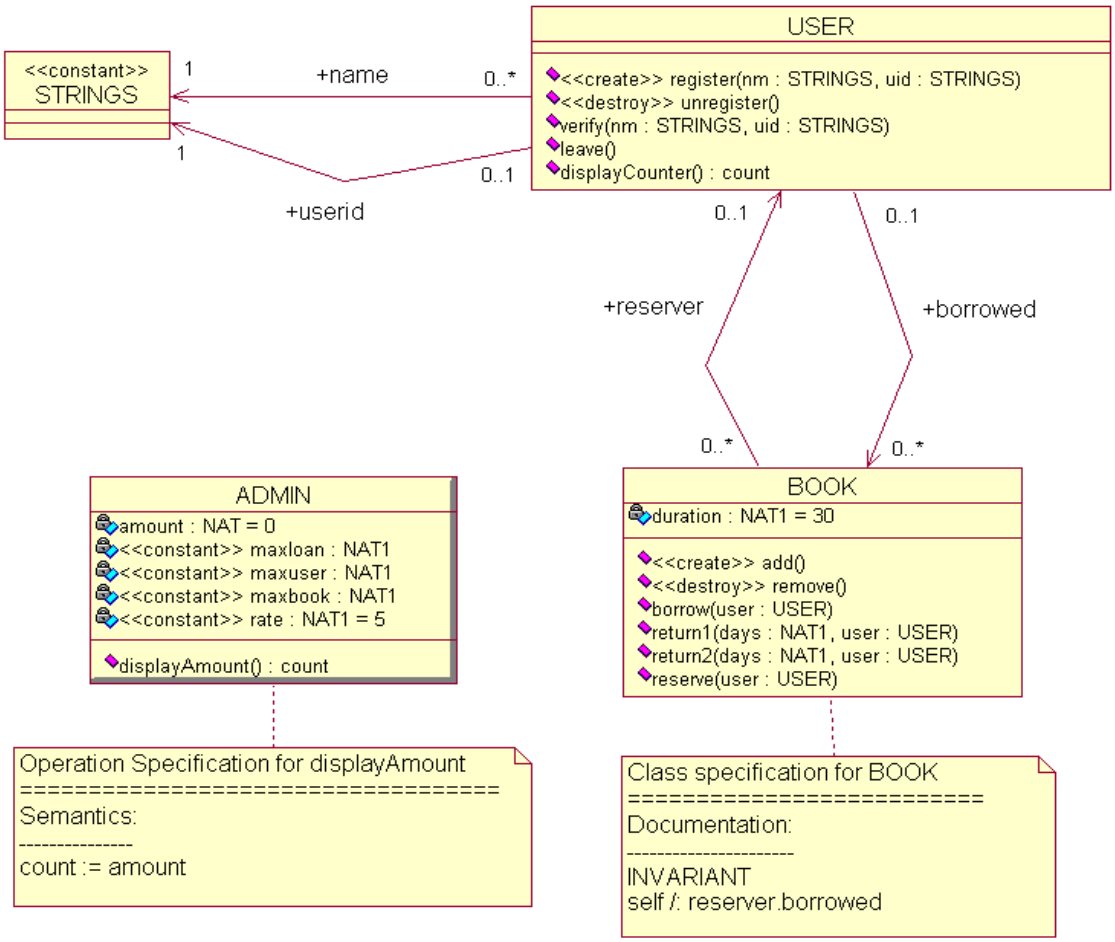


Statechart Diagram for AUCTION Class

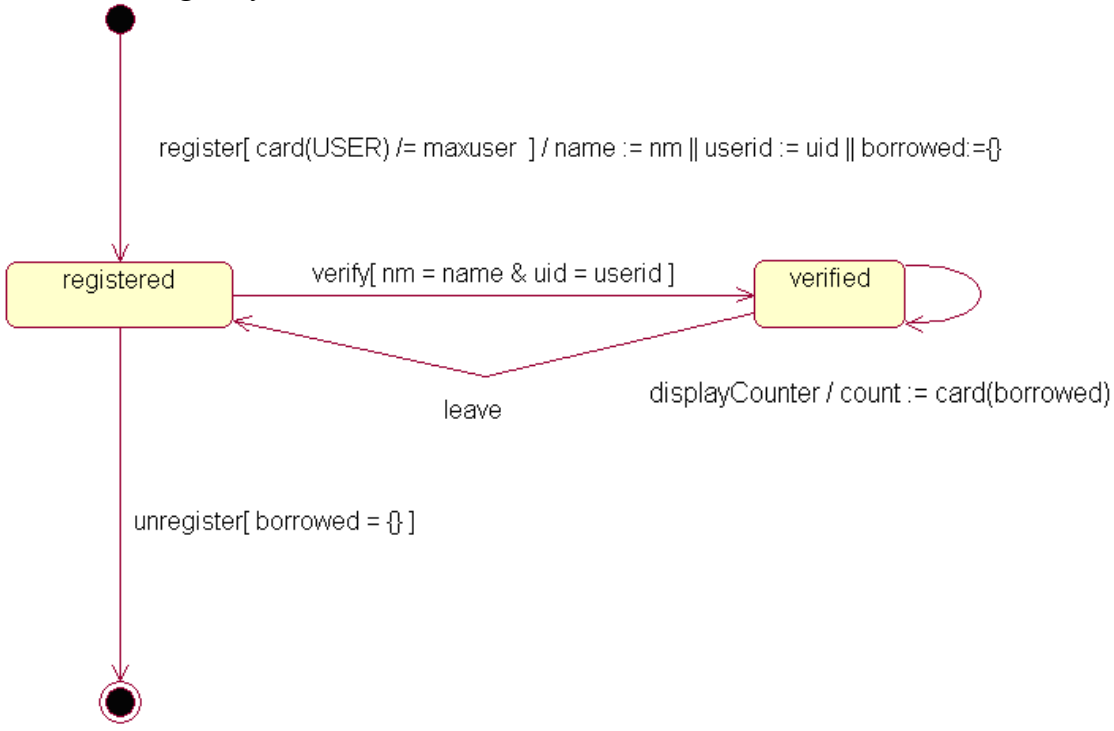


Case 2: Library System

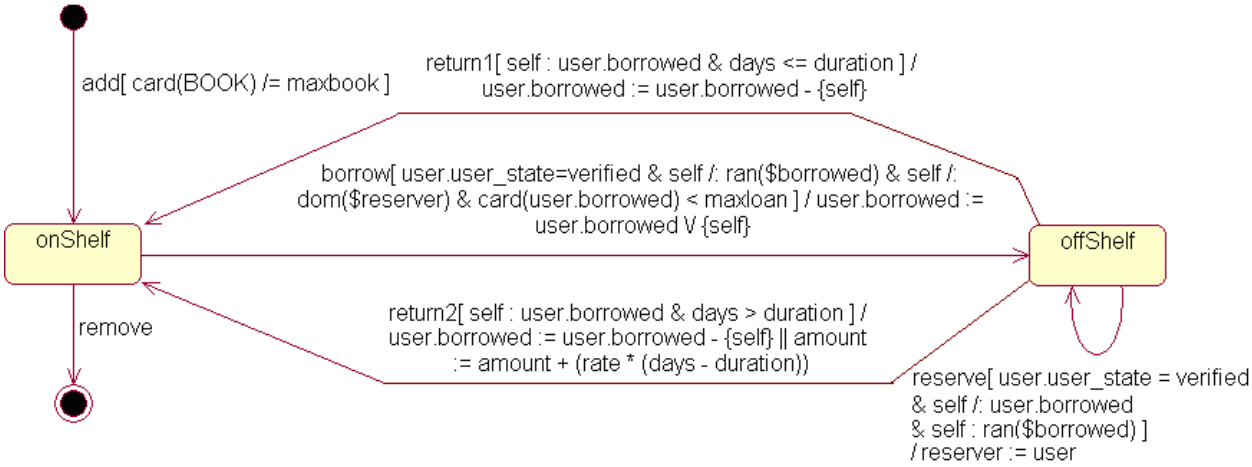
Class Diagram



Statechart Diagram for USER Class



Statechart Diagram for BOOK Class



A.1.2. B Models

Case 1: Auction System

```
MACHINE Auction

CONSTANTS    commission

PROPERTIES   commission: NAT1

SETS
        AUCTION;
        STRINGS;
        USER;
        AUCTION_STATE={noBids,bidding};
        USER_STATE={loggedOut,loggedIn}

VARIABLES
        auction,
        registered,
        reserve,
        highest_bid,
        seller,
        highest_bidder,
        balance,
        name,
        password,
        admin_amount,
        auction_state,
        user_stat

INVARIANT
        auction : POW(AUCTION) &
        registered : POW(USER) &
        reserve : auction --> NAT1 &
        highest_bid : auction --> NAT &
        seller : auction --> registered &
```

```
        highest_bidder : auction +-> registered &
        balance : registered --> NAT &
        name : registered >-> STRINGS &
        password : registered --> STRINGS &
        admin_amount : NAT &
        auction_state : auction --> AUCTION_STATE &
        user_state : registered --> USER_STATE &
        !(aa).(aa : auction => (seller(aa) /= highest_bidder(aa) ))

INITIALISATION
        auction :={} ||
        registered :={} ||
        reserve := {} ||
        highest_bid := {} ||
        seller := {} ||
        highest_bidder := {} ||
        balance := {} ||
        name := {} ||
        password := {} ||
        admin_amount := 0 ||
        auction_state := {} ||
        user_state := {}

OPERATIONS

createAuction(uu,aa,rr) =
PRE uu : USER & aa : AUCTION & rr : NAT1
THEN
        SELECT
                aa : AUCTION - auction &
                commission <= balance(uu) &
                user_state(uu) = loggedIn
        THEN
                auction := auction \/ {aa} ||
                seller(aa) := uu ||
                reserve(aa) := rr ||
                balance(uu) := balance(uu) - commission ||
                admin_amount := admin_amount + commission ||
                highest_bid(aa) :=0 ||
                auction_state(aa) :=noBids
        END
END
;

placeBid1(uu,aa,bb) =
PRE uu : USER & aa : AUCTION & bb : NAT1
THEN
        SELECT
                aa : auction &
                aa /: dom(highest_bidder) &
                uu /= seller(aa) &
                bb >= reserve(aa) &
                bb <= balance(uu) &
                highest_bid(aa) = 0 &
                auction_state(aa) = noBids &
                user_state(uu) = loggedIn
        THEN
                highest_bid(aa) := bb ||
                highest_bidder(aa) := uu ||
                balance(uu) := balance(uu) - bb ||
```

```
                auction_state(aa) := bidding
        END
END
;

placeBid2(uu,aa,bb) =
PRE uu : USER & aa : AUCTION & bb : NAT1
THEN
        SELECT
                aa : auction &
                aa : dom(highest_bidder) &
                uu /= seller(aa) &
                uu /= highest_bidder(aa) &
                bb > highest_bid(aa) &
                bb <= balance(uu) &
                auction_state(aa) = bidding &
                user_state(uu) = loggedIn
        THEN
                highest_bidder(aa) := uu ||
                highest_bid(aa) := bb ||
                balance := balance <+ {uu |-> balance(uu) - bid , highest_bidder(aa) |->
                balance(highest_bidder(aa)) + highest_bid(aa)}
        END
END
;

win(uu,aa) =
PRE uu : USER & aa : AUCTION
THEN
        SELECT
                aa : auction &
                uu = highest_bidder(aa) &
                highest_bid(aa) >= reserve(aa) &
                commission <= balance(seller(aa)) + highest_bid(aa) &
                auction_state(aa) = bidding
        THEN
                auction := auction - {aa} ||
                balance(seller(aa)) := balance(seller(aa)) + highest_bid(aa) –
                commission ||
                admin_amount := admin_amount + commission ||
                reserve := {aa} <<| reserve ||
                highest_bid := {aa} <<| highest_bid ||
                seller := {aa} <<| seller ||
                highest_bidder := {aa} <<| highest_bidder ||
                auction_state := {aa} <<| auction_state
        END
END
;

register(uu,nn,pp) =
PRE uu : USER & nn : STRINGS & pp : STRINGS
THEN
        SELECT
                uu : USER – registered &
                nn /: ran(name)
        THEN
                registered := registered \/ {uu} ||
                name(uu) := nn ||
                password(uu) := pp ||
                balance(uu) := 0 ||
```

```
                user_state(uu) := loggedOut
        END
END
;

unregister(uu) =
PRE uu : USER
THEN
        SELECT
                uu /: ran(seller) &
                uu /: ran(highest_bidder) &
                balance(uu) = 0 &
                user_state(uu)=loggedIn
        THEN
                registered := registered - {uu} ||
                name := {uu} <<| name ||
                password := {uu} <<| password ||
                balance := {uu} <<| balance ||
                user_state := {uu} <<| user_state
        END
END
;

login(uu,nn,pp) =
PRE uu : USER & nn : STRINGS & pp : STRINGS
THEN
        SELECT
                nn = name(uu) &
                pp = password(uu) &
                user_state(uu) = loggedOut
        THEN
                user_state(uu):=loggedIn
        END
END
;

logoff(uu) =
PRE uu : USER
THEN
        SELECT
                user_state(uu)=loggedIn
        THEN
                user_state(uu):=loggedOut
        END
END
;

amount <--  displayAmount(uu) =
PRE uu : USER
THEN
        SELECT
            user_state(uu)=loggedIn
        THEN
            amount:=balance(uu)
        END
END
;
```

```
amount <--  displayAdminAmount =
BEGIN
        amount:=admin_amount
END

END
```

Case 2: Library System

```
MACHINE Library

CONSTANTS
        maxloan,
        maxuser,
        maxbook,
        rate


PROPERTIES
        maxloan : NAT1 &
        maxuser : NAT1 &
        maxbook : NAT1 &
        rate : NAT1 &
        rate = 5


SETS
        BOOK;
        STRINGS;
        USER;
        BOOK_STATE={onShelf,offShelf};
        USER_STATE={registered,verified}


VARIABLES
        book,
        user,
        duration,
        reserver,
        borrowed,
        name,
        userid,
        amount,
        book_state,
        user_state


INVARIANT
        book:POW(BOOK) &
        user:POW(USER) &
        duration : book --> NAT1 &
        reserver : book +-> user &
        borrowed : user -->POW(book) &
        name : user >-> STRINGS &
        userid : user --> STRINGS &
        amount : NAT &
        book_state : book --> BOOK_STATE &
```

```
        user_state : user --> USER_STATE &
        !(bb).(bb:book => (bb /: borrowed(reserver(bb)))) &
        !(a1,a2).( a1:dom(borrowed) & a2:dom(borrowed) & a1/=a2 => borrowed(a1) /\
        borrowed (a2)={} )

INITIALISATION
        book := {} ||
        user := {} ||
        duration := {} ||
        reserver := {} ||
        borrowed := {} ||
        name := {} ||
        userid := {} ||
        amount := 0 ||
        book_state := {} ||
        user_state := {}


OPERATIONS
add(bb) =
PRE bb : BOOK
THEN
        SELECT
                bb : BOOK - book &
                card(book) /= maxbook
        THEN
                book := book \/ {bb} ||
                book_state(bb) := onShelf ||
                duration(bb) := 30

        END
END
;

remove(bb) =
PRE bb : BOOK
THEN
        SELECT
                book_state(bb) = onShelf
        THEN
                book := book - {bb} ||
                book_state := {bb} <<| book_state ||
                duration := {bb} <<| duration ||
                reserver := {bb} <<| reserver

        END
END
;

borrow(bb,uu) =
PRE bb : BOOK & uu : USER
THEN
        SELECT
                user_state(uu) = verified &
                bb /: ran(borrowed) &
                bb /: dom(reserver) &
                card(borrowed(uu)) < maxloan &
                book_state(bb) = onShelf
        THEN
                book_state(bb) := offShelf ||
```

```
                borrowed(uu) := borrowed(uu) \/ {bb}

        END
END;

return1(bb,uu,dd) =
PRE bb : BOOK & uu : USER & dd : NAT1
THEN
        SELECT
                bb : borrowed(uu) &
                dd <= duration(bb) &
                book_state(bb)=offShelf
        THEN
                book_state(bb):=onShelf ||
                borrowed(uu) := borrowed(uu) - {bb}

        END
END
;

return2(bb,uu,dd) =
PRE bb : BOOK & uu : USER & dd : NAT1
THEN
        SELECT
                bb : borrowed(uu) &
                dd > duration(bb) &
                book_state(bb)=offShelf
        THEN
                book_state(bb):=onShelf ||
                borrowed(uu) := borrowed(uu) - {bb}||
                amount := amount + (rate * (dd - duration(bb)))

        END
END
;

reserve(bb,uu) =
PRE bb : BOOK & uu : USER
THEN
        SELECT
                user_state(uu) = verified &
                bb /: borrowed(uu) &
                bb : ran(borrowed) &
                book_state(bb)=offShelf
        THEN
                reserver(bb) := uu

        END
END
;

register(uu,nn,ii) =
PRE uu : USER & nn : STRINGS & ii : STRINGS
THEN
        SELECT
                uu : USER – user &
                card(user) /= maxuser
        THEN
                user := user \/ {uu} ||
                user_state(uu) := registered ||
```

```
                    name(uu) := nn ||
                    userid(uu) := ii ||
                    borrowed(uu) := {}

        END
END;

unregister(uu) =
PRE uu : USER
THEN
        SELECT
                uu /: ran(borrowed) &
                user_state(uu) = registered &
                borrowed(uu) = {}
        THEN
                user := user - {uu} ||
                user_state := {uu} <<| user_state ||
                name := {uu} <<| name ||
                userid := {uu} <<| userid ||
                borrowed := {uu} <<| borrowed

        END
END
;

verify(uu,nn,ii) =
PRE uu : USER & nn : STRINGS & ii : STRINGS
THEN
        SELECT
                nn = name(uu) &
                ii = userid(uu) &
                user_state(uu) = registered
        THEN
                user_state(uu) := verified
        END
END;

leave(uu) =
PRE uu : USER
THEN
        SELECT
                user_state(uu) = verified
        THEN
                user_state(uu) := registered
        END
END
;

count <--  displayCounter(uu) =
PRE uu : USER
THEN
        SELECT
                user_state(uu) = verified
        THEN
                count := card(borrowed(uu))
        END
END
;
```

```
count <-- displayAmount =
BEGIN
        count := amount
END
END
```

A.2. Questions

A.2.1. UML-B Models Questionnaire

Case 1: Auction System

1. What does the **.** (dot) symbol between entities (for example, **user.balance)** mean in the UML-B model?

2. In your own words (i.e. natural language), describe the difference in output (i.e. **amount**) between **displayAmount** and **displayAdminAmount** operations.

3. By analysing the UML-B model, is it possible for a user to place a bid on his/her own created auction and be the highest bidder of that auction? Circle the answer.

   Yes **No** **Not Sure**

   If **Yes** or **No**, which part(s) of the model do(es) indicate you this? If **Not Sure**, why?

4. In your own words (i.e. natural language), describe the difference in functionality between **placeBid1** and **placeBid2** operations.

   What does the following statement of **placeBid2** operation mean exactly in natural language? (4 marks)

   $balance:= $balance <+ {user |-> user.balance - bid , highest_bidder |-> highest_bidder.balance + highest_bid}

5. A new requirement is added to the system:

   A user can change his/her password, where he/she provides his/her current password and proposes a new password. The new password must be different from the current one. The number of times the password has been changed must not exceed the maximum number of times allowed. He/she must log in for this operation.

   How would you introduce/add this requirement to the current UML-B model?
   Formulate your solution explicitly by sketching the necessary elements on the given UML-B model sheets (i.e. class diagram and statechart diagrams).

Case 2: Library System

1. What does the **$** symbol preceding an entity (for example, **$borrowed**) mean in the UML-B model's semantics?

2. In your own words (i.e. natural language), describe the difference in output (i.e. **count**) between **displayAmount** and **displayCounter** operations.

3. By analysing the UML-B model, is it possible for a book to be reserved by the same user who is currently borrowing it? Circle the answer.

Yes **No** **Not Sure**

If **Yes** or **No**, which part(s) of the model do(es) indicate you this? If **Not Sure**, why?

4. In your own words (i.e. natural language), describe the difference in functionality between **return1** and **return2** operations.

   What does the following statement of **return2** operation mean exactly in natural language?

   amount:= amount + (rate * (days – duration))

5. A new requirement is added to the system:

   A user can renew a book provided another user has not reserved the book. The number of times the book has been renewed by him/her must not exceed the maximum number of renewal allowed. His/her identification must be verified for this operation.

   How would you introduce/add this requirement to the current UML-B model?

   Formulate your solution explicitly by sketching the necessary elements on the given UML-B model sheets (i.e. class diagram and statechart diagrams).

A.2.2. B Models Questionnaire

Case 1: Auction System

1. What does a parenthesised identifier mean when it follows a reference to a variable name (for example, **balance(uu)**) in the B model?

2. In your own words (i.e. natural language), describe the difference in output (i.e. **amount**) between **displayAmount** and **displayAdminAmount** operations.

3. By analysing the B model, is it possible for a user to place a bid on his/her own created auction and be the highest bidder of that auction? Circle the answer.

   Yes **No** **Not Sure**

   If **Yes** or **No**, which part(s) of the model do(es) indicate you this? If **Not Sure**, why?

4. In your own words (i.e. natural language), describe the difference in functionality between **placeBid1** and **placeBid2** operations.

   What does the following statement of **placeBid2** operation mean exactly in natural language?

   balance:= balance <+ {uu |-> balance(uu) - bid , highest_bidder(aa) |-> balance(highest_bidder(aa)) + highest_bid(aa)}

5. A new requirement is added to the system:

   A user can change his/her password, where he/she provides his/her current password and proposes a new password. The new password must be different from the current one. The number of times the password has been changed must not exceed the maximum number of times allowed. He/she must log in for this operation.

   How would you introduce/add this requirement to the current B model?

Formulate your solution explicitly by adding the necessary elements on the given B model sheets.

Case 2: Library System

1. What does the **!** symbol preceding an entity (for example, **!(bb)**) mean in the B model?

2. In your own words (i.e. natural language), describe the difference in output (i.e. **count**) between **displayAmount** and **displayCounter** operations.

3. By analysing the B model, is it possible for a book to be reserved by the same user who is currently borrowing it? Circle the answer.

   Yes **No** **Not Sure**

   If **Yes** or **No**, which part(s) of the model do(es) indicate you this? If **Not Sure**, why?

4. In your own words (i.e. natural language), describe the difference in functionality between **return1** and **return2** operations.

   What does the following statement of **return2** operation mean exactly in natural language?

   amount:= amount + (rate * (days – duration))

5. A new requirement is added to the system:

   A user can renew a book provided another user has not reserved the book. The number of times the book has been renewed by him/her must not exceed the maximum number of renewal allowed. His/her identification must be verified for this operation.

   How would you introduce/add this requirement to the current B model?

   Formulate your solution explicitly by adding the necessary elements on the given B model sheets.

A.2.3. Debriefing Questionnaire

Strategies in Answering Questions

Please **tick one or more** boxes that best describe your answer.

How did you derive the answer for **Question 1..5**? Was it based on:

A. Guess or Hunch	B. Domain Knowledge or Previous Knowledge	C. Logic or Common Sense	D. Reading and understanding the model	E. Other

If **Other**, please specify:

Breadth of Comprehension Strategies

Read the following strategies.

Strategy A
I analysed the model in detail by identifying the roles of attributes and operations in all classes. I also traced through the relationships and dependencies between interacted classes in order to gain a global understanding of the model. The intention was to understand the overall design so that I could make the modification fits safely with the existing design.

Strategy B
I focused only on the selected parts of the model relating to the modification task that I needed to do. In other words, I attempted to understand the minimum amount of information from the model, which I believed is necessary to successfully carry out the modification task.

When answering **Question 5**, which strategy did you use? Was it **Strategy A** or **Strategy B** or **Both** or **Other**? If **Other**, please specify.

Direction of Comprehension Strategies

Read the following strategies.

Strategy A
I began by firstly making a general assumption about the system described by the model based on the information available at first glance (i.e. class titles, model structure etc.). The assumption leaded me to expect certain classes, attributes and operations in the model, which resulted in another level or more specific assumptions. I later continued looking for specific information in the model to verify (i.e. accept or reject) the assumptions.

Strategy B
I began by firstly analysing specific attribute(s) or operation(s) of a class. I understood the functionality of that class locally. That individual class was later linked with other related classes in order to understand the relationships between these classes and their dependencies. In this way, I obtained a unit of information about the model. Later, I repeated the process until the whole model was reviewed and understood.

In general, which strategy did you use to comprehend the model? Was it **Strategy A** or **Strategy B** or **Both** or **Other**? If **Other**, please specify.

Model Preference and Comments on the Models

Please answer the following questions as sincerely as possible. Your answers will be treated confidentially.

Having done some exercises on both models:

1. How would you rate the UML-B model comprehensibility? Circle the answer.

Very difficult to comprehend				**Very easy to comprehend**
-2	**-1**	**0**	**1**	**2**

What is the hardest part to understand about the UML-B model?

2. How would you rate the B model comprehensibility? Circle the answer.

Very difficult to comprehend				**Very easy to comprehend**
-2	**-1**	**0**	**1**	**2**

What is the hardest part to understand about the B model?

3. If you are given the choice, which model would you prefer to deal with: **UML-B** or **B**? Why?

4. Any other comments (i.e. positive and/or negative feedback)?

A.3. Raw Data

Case 1: Auction System

Subject	Overall Comprehension Task	Comprehension for Modification Task
Case 1:UML-B (Unit: Marks/min)		
X1	0.93	1.14
X2	0.34	N/A
X3	0.76	1.22
X4	1.03	1.88
X5	0.75	0.53
X6	0.70	1.00
X7	1.00	1.75
X8	0.63	1.00
X9	1.26	2.00
X10	1.33	1.88
X11	1.11	1.50
X12	0.63	1.00
X13	0.64	0.67
X14	0.13	0.00
X15	0.29	0.00
X16	0.50	1.20
X17	0.28	N/A
X18	0.66	1.38
X19	0.59	N/A
X20	1.13	1.50
X21	0.85	2.00
Case 1: B (Unit: Marks/min)		
Y1	0.42	0.00
Y2	0.87	1.50
Y3	0.96	0.83
Y4	0.39	0.00
Y5	0.76	0.92
Y6	0.88	1.71
Y7	1.12	1.00
Y8	0.40	0.42
Y9	0.26	N/A
Y10	0.83	2.00
Y11	0.48	0.09
Y12	0.41	N/A

Y13	0.68	0.50
Y14	0.63	0.50
Y15	0.17	N/A
Y16	0.42	N/A
Y17	0.67	1.75
Y18	0.24	0.40
Y19	0.63	0.58
Y20	0.71	0.58

Case 2: Library System

Subject	Overall Comprehension Task	Comprehension for Modification Task
Case 2:UML-B (Unit: Marks/min)		
Y1	0.87	1.33
Y2	0.96	0.67
Y3	0.85	0.33
Y4	0.74	0.67
Y5	0.86	0.73
Y6	1.14	0.64
Y7	0.74	0.47
Y8	0.71	0.41
Y9	0.90	1.00
Y10	0.63	0.46
Y11	0.57	0.63
Y12	0.69	1.50
Y13	0.75	0.63
Y14	1.06	1.60
Y15	0.63	N/A
Y16	0.52	0.40
Y17	0.76	0.90
Y18	0.28	0.40
Y19	0.69	0.46
Y20	0.77	1.27
Case 2: B (Unit: Marks/min)		
X1	1.04	0.91
X2	0.71	0.64
X3	0.51	0.50
X4	0.94	0.75
X5	0.63	0.39
X6	0.70	0.30
X7	1.18	0.67
X8	0.61	0.25
X9	1.00	1.11
X10	0.91	0.90
X11	0.71	0.50
X12	0.53	0.40
X13	0.73	0.50
X14	0.50	0.00
X15	0.43	0.86
X16	0.44	0.20
X17	0.57	0.45
X18	0.48	0.18
X19	0.88	0.60
X20	1.08	1.10
X21	0.71	1.20

Appendix B

Survey 1

This appendix presents the questions used in the first survey (**Chapter 4**).

Questions:

(1) If you need to compare different parts of your **UML-B model** (e.g. between diagrams or windows of different operations etc.), how easy is it to view them at the same time in **Rose**?

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

Why?

(2) If you need to rebuild/restructure your **UML-B model** (e.g. due to change in ideas or requirements or solutions), how easy is it to make the changes?

Note: The changes include editing the diagrams and the respective semantics in **Rose** and retranslating the model to B model using **U2B**.

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

Are there any particular changes that are particularly difficult or tedious to make?

Yes **No** **Not Sure**

If **Yes**, which ones?

(3) How simple is it to describe what you intend when modeling your **UML-B model**?

Very complicated			**Very Simple**	
-2	**-1**	**0**	**1**	**2**

Why?

(4)

(i) How easy is it to make mistakes when modeling the diagrams in your **UML-B model**?

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

Why?

(ii) How easy is it to make mistakes when defining the formal semantics in microB clauses for the diagrams in your **UML-B model**?

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

Why?

(5) Can you stop modeling your **UML-B model** at any time you like and check your work so far (i.e. by translating it to B model using **U2B** and performing model validation and verification in **ProB**)?

Yes **No** **Not Sure**

If **No**, why?

(6) Do you find any complex or difficult tasks to work out in your head when modeling your **UML-B model**?

Yes **No** **Not Sure**

If **Yes**, what are they?

(7) Are there any parts in the **UML-B model** that seem to be similar in functionality but the **UML-B method** makes them appear different?

Yes **No** **Not Sure**

If **Yes**, what are they?

(8) Do you find any structure dependencies in **UML-B model** (i.e. one part explicitly relies upon or is determined by or uses or requires another part)?

Yes **No** **Not Sure**

If **Yes**, how visible are the structure dependencies?

(a) Visible in both parts
(b) Visible in one part
(c) Not visible in both parts

If **(b)** or **(c)**, what are the parts involved? Please state them specifically.

(9) Does **Rose** allow you to make notes or convey extra information beyond the **UML-B model** to yourself (e.g. comments, use different fonts, layout)?

Yes **No** **Not Sure**

If **Yes**, please state the possible actions.

(10)

(i) How easy is it to determine what each diagram (and its components) is for in the **UML-B model** as a whole?

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

Why?

(ii) How easy is it to determine what each microB clause is for in the **UML-B model** as a whole?

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

Why?

(iii) Are there any parts that you simply include just because it is always been that way (without exactly knowing what the purposes)?

Yes **No** **Not Sure**

If **Yes**, what are they?

(11) How well does the **UML-B method** allow you to describe your problem accurately and completely as what you intend?

Very Bad			**Very Good**	
-2	**-1**	**0**	**1**	**2**

Why?

(12) How well does the **UML-B method** allow you to play around with your model (e.g. when you are testing your ideas/solutions, without being sure what the effects will be)?

Very Bad			**Very Good**	
-2	**-1**	**0**	**1**	**2**

Which part of the method help or prevent you to do this?

(13) Can you go about any task in any order you like in the **UML-B method**?

Yes **No** **Not Sure**

If **No**, why?

(14) Does the **UML-B method** insist you start the modeling task by defining or grouping things first before you can do anything else?

Yes **No** **Not Sure**

If **Yes**, what sort of things?

(15) How easy is it to learn the **UML-B method** compared to the traditional **B method**?

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

Are there any particular parts in the **UML-B method** that are particularly difficult to learn and understand how they work?

Yes **No** **Not Sure**

If **Yes**, which ones?

(16)

(i) How easy is it to learn and use the **U2B** tool?

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

Why?

(ii) Has the tool met its purpose and your expectation (i.e. is it useful)?

Yes, a lot **Yes** **Yes, a little** **Not Sure** **No**

Why?

(17) How useful do you find the available manual and documentation on the **UML-B method**?

Very Useless				**Very Useful**
-2	**-1**	**0**	**1**	**2**

Why?

(18) How easy is it to become familiar with the **UML-B method** and be able to use it in your task efficiently without referring to the documentation?

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

Why?

(19) How easy is it to do modeling using the **UML-B method** compared to the traditional **B method**?

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

If you are given the choice in modeling, which method would you choose: **UML-B** or **B**? Why?

(20) Do you find yourself using the **UML-B method** and **U2B** in ways that are unusual or ways that the method designer might not have intended? Can you think of obvious ways that the design of the **UML-B method** and **U2B** could be improved? What are they?

Appendix C

Controlled Experiment 2

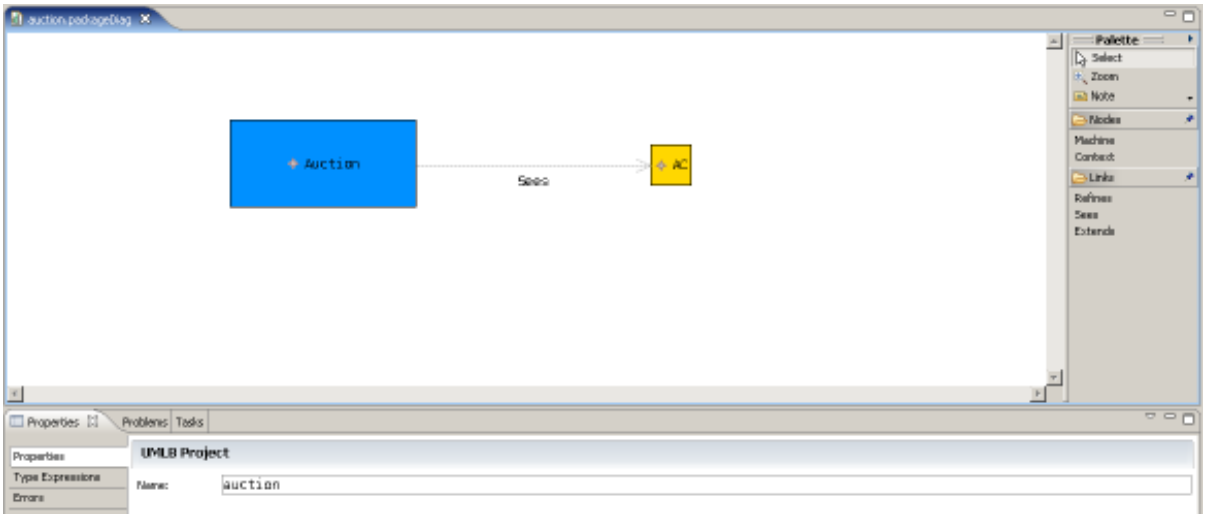
This appendix presents the materials used in the second controlled experiment (**Chapter 5**). This includes the models and questions used in the experiment.

C.1. Models

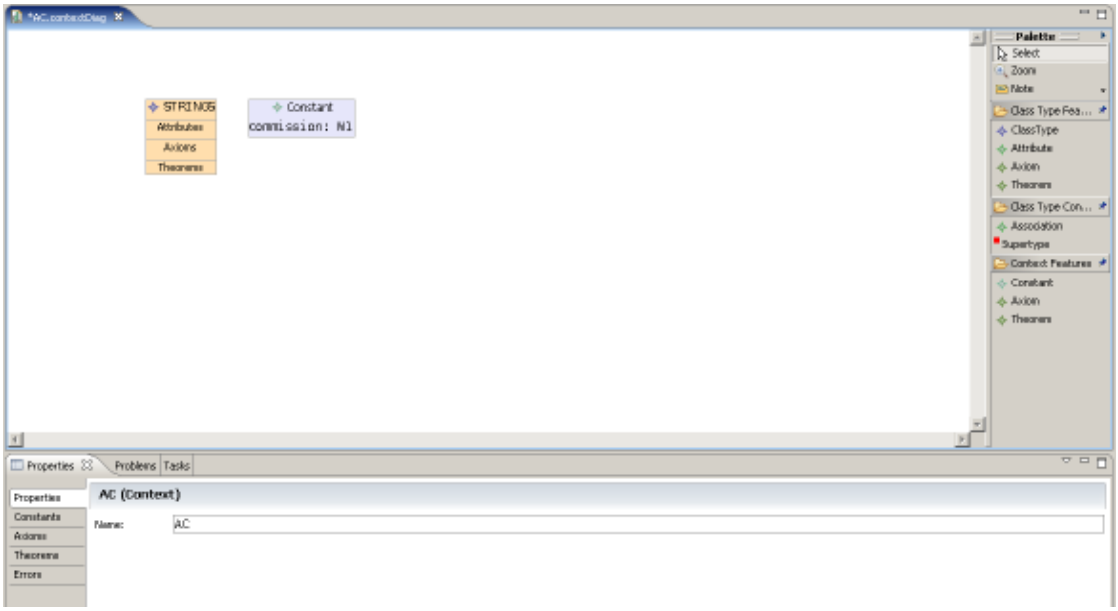
C.1.1. UML-B Models

Case 1: Auction System

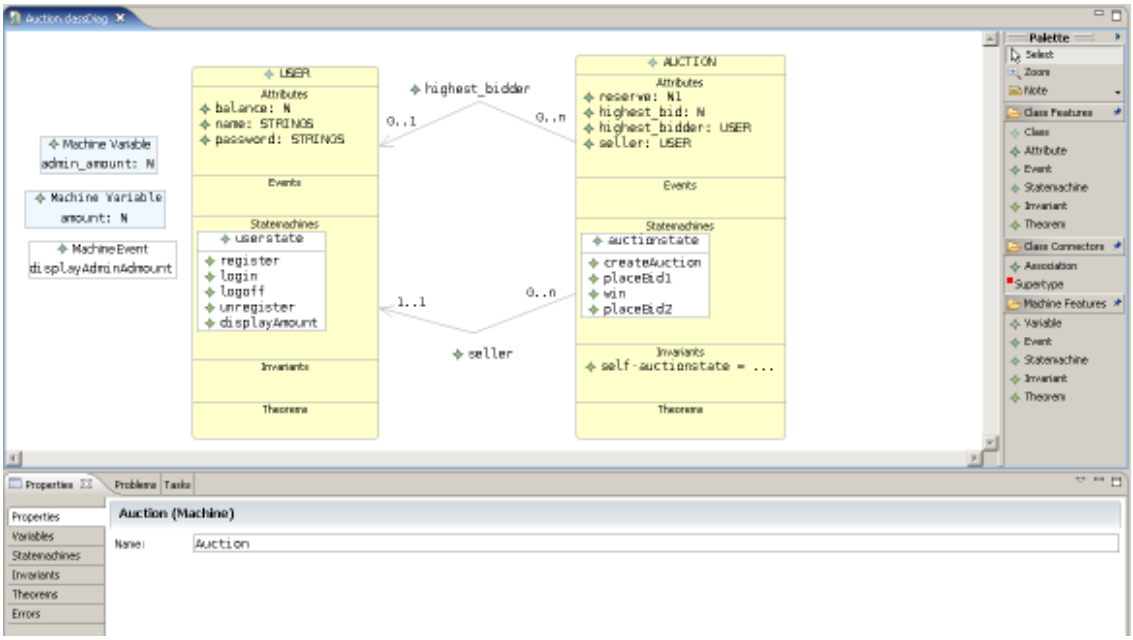
Package Diagram



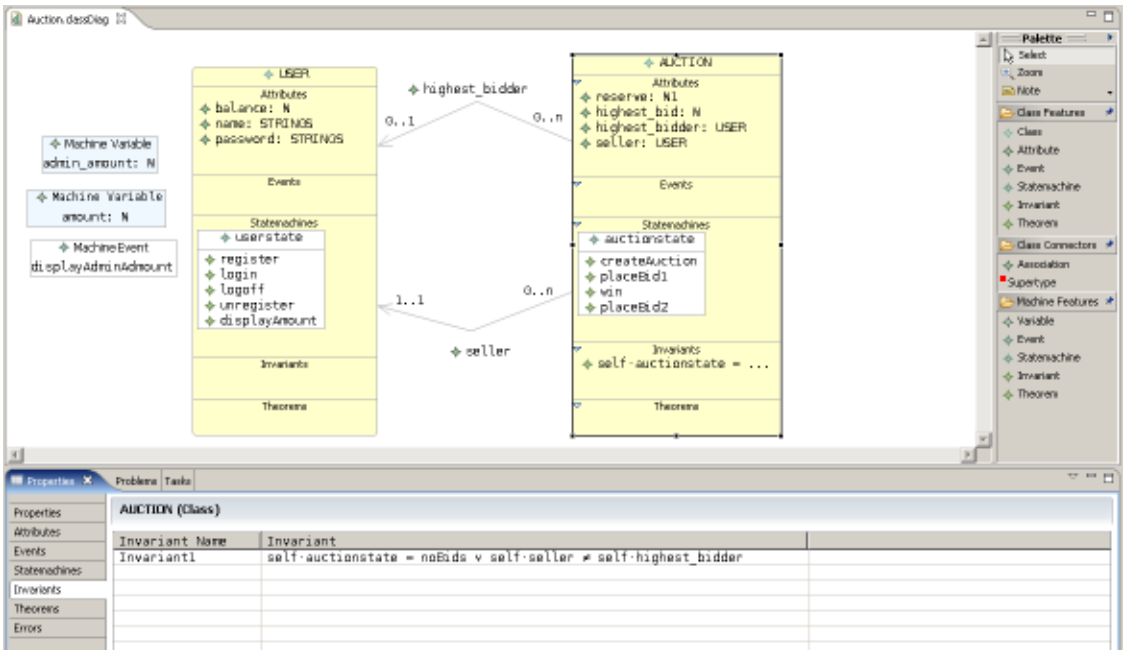
Context Diagram



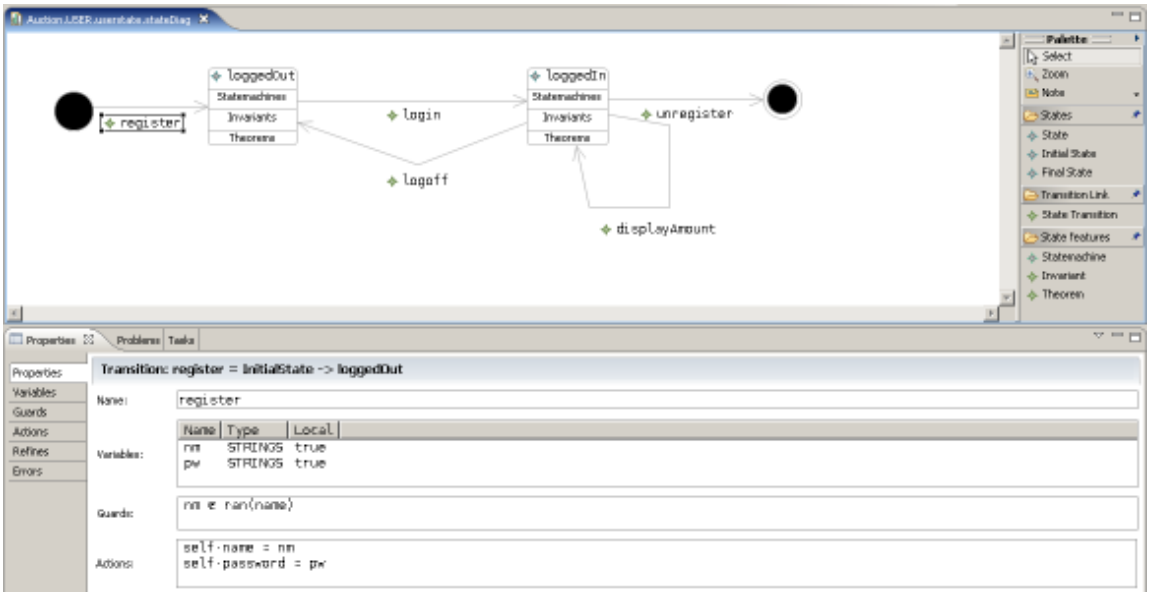
Class Diagram (Machine)



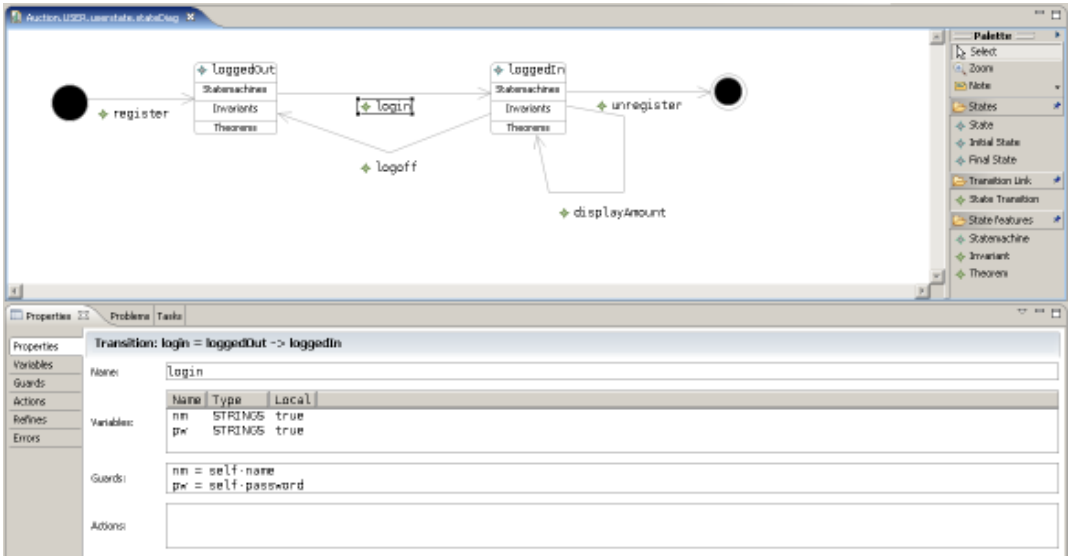
Class Diagram (Machine) - Continued

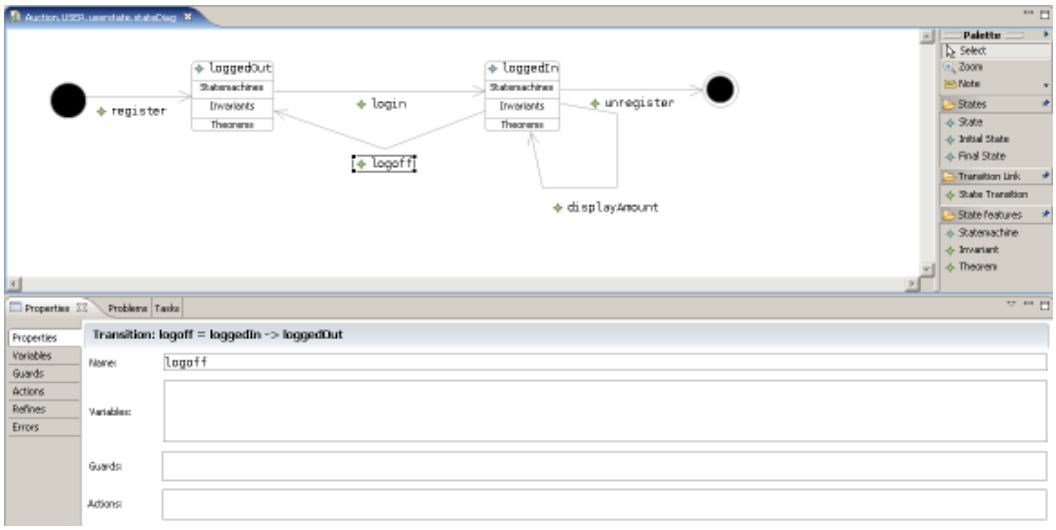


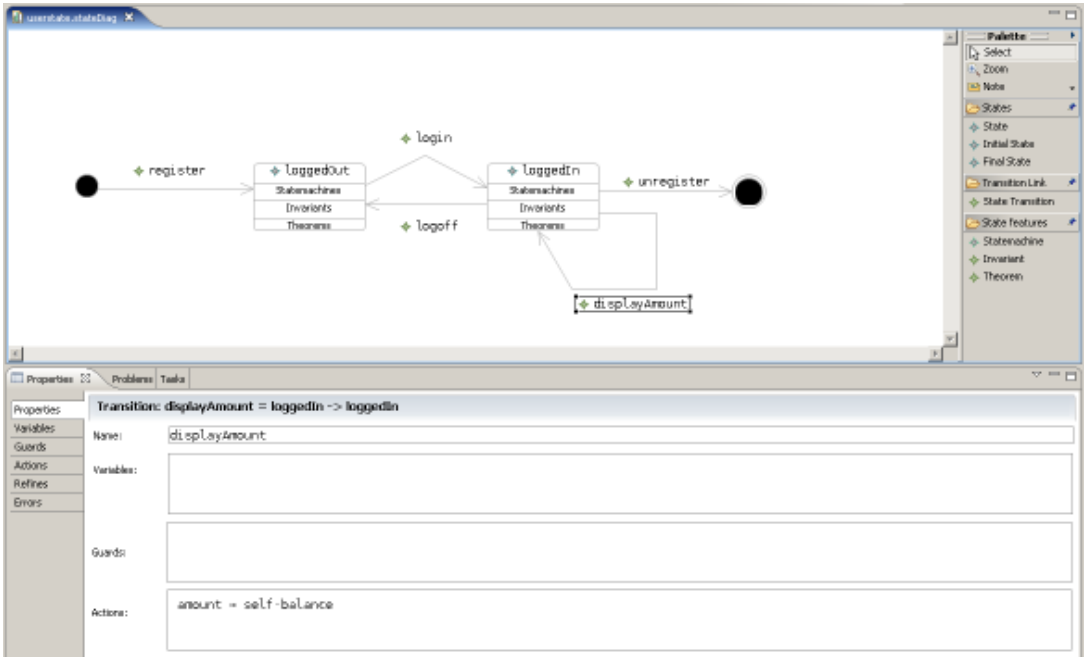
State Diagram for USER Class



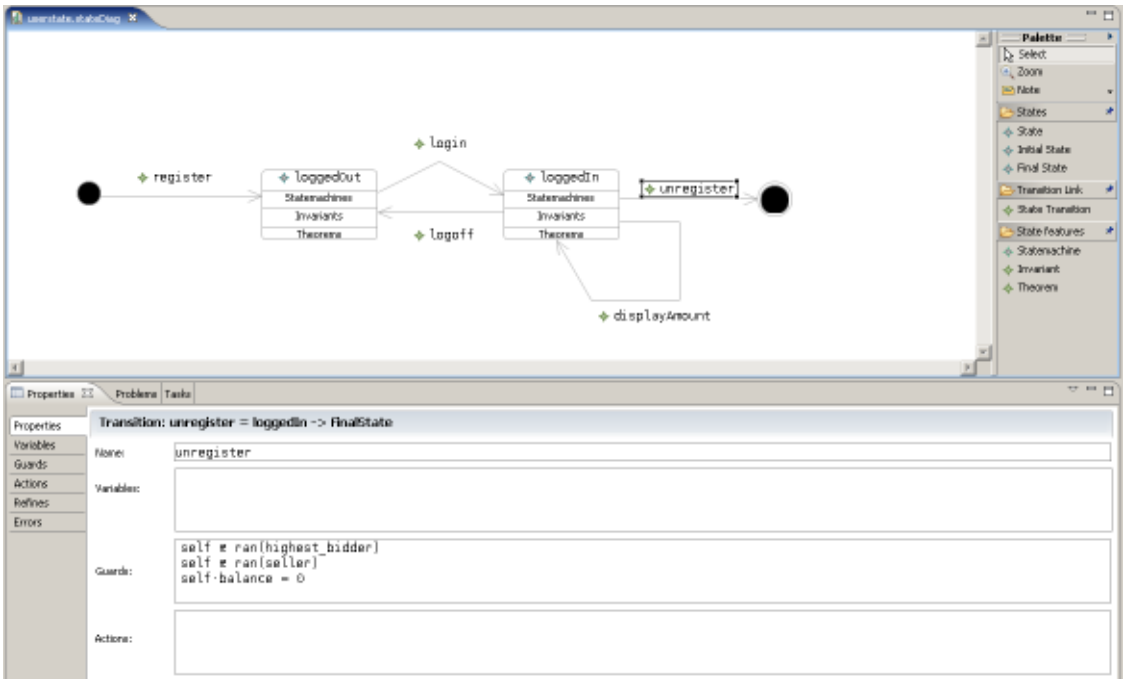
State Diagram for USER Class - Continued



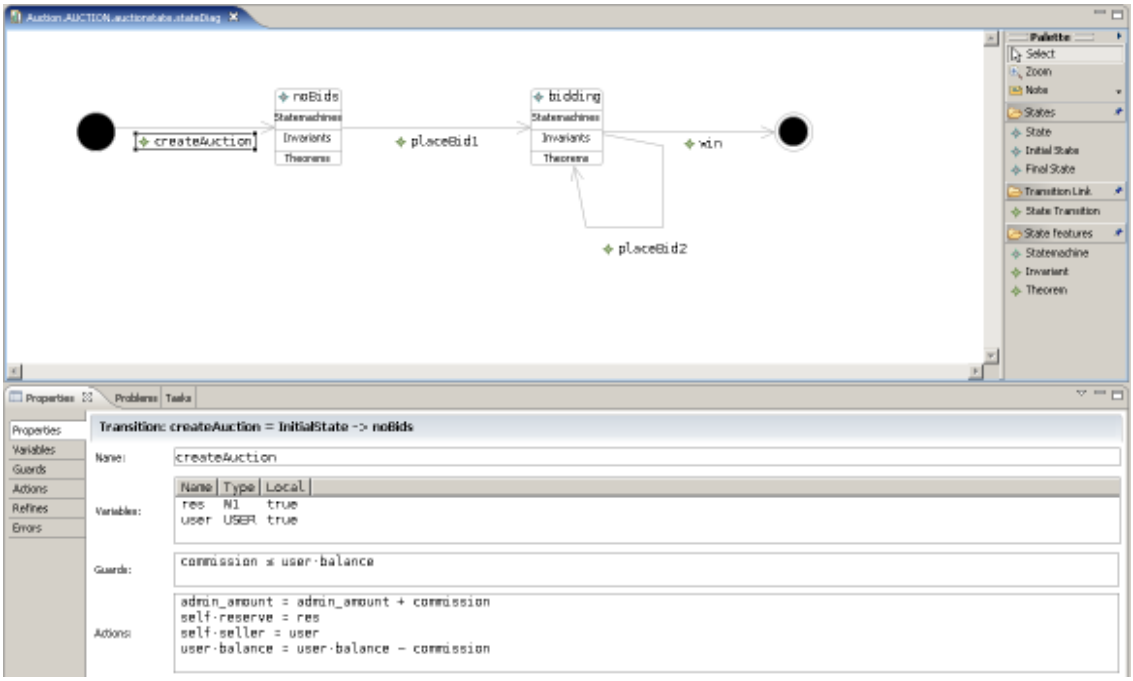




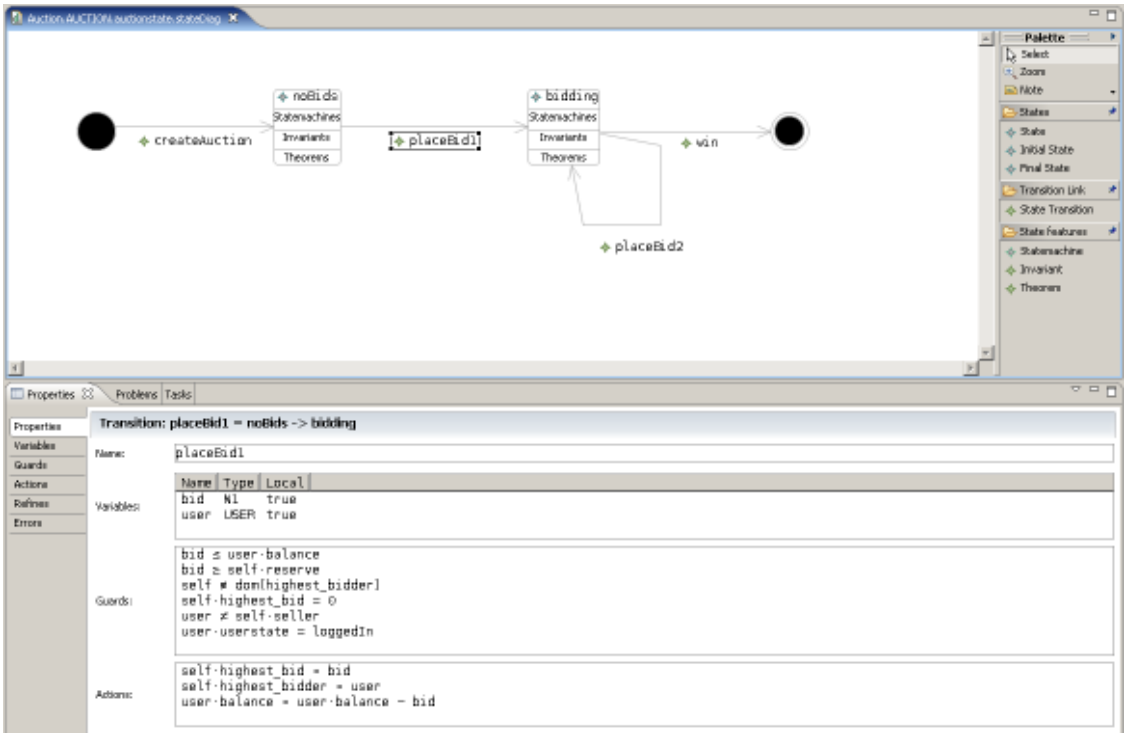
State Diagram for USER Class - Continued

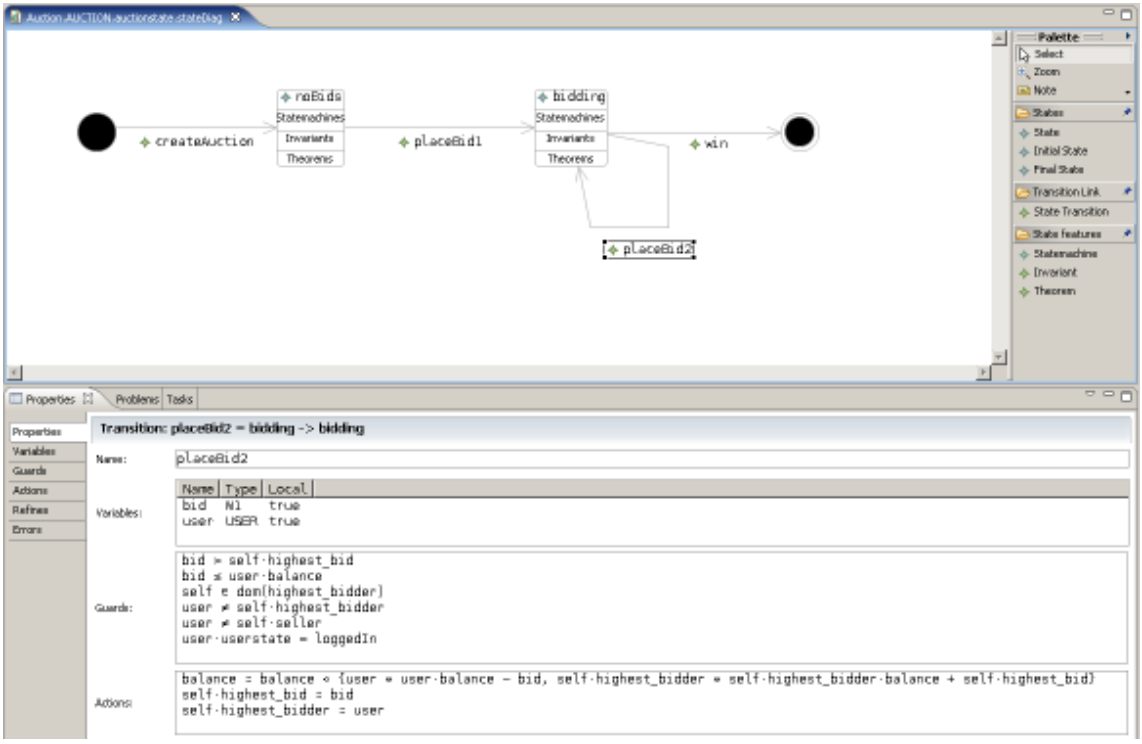


State Diagram for AUCTION Class

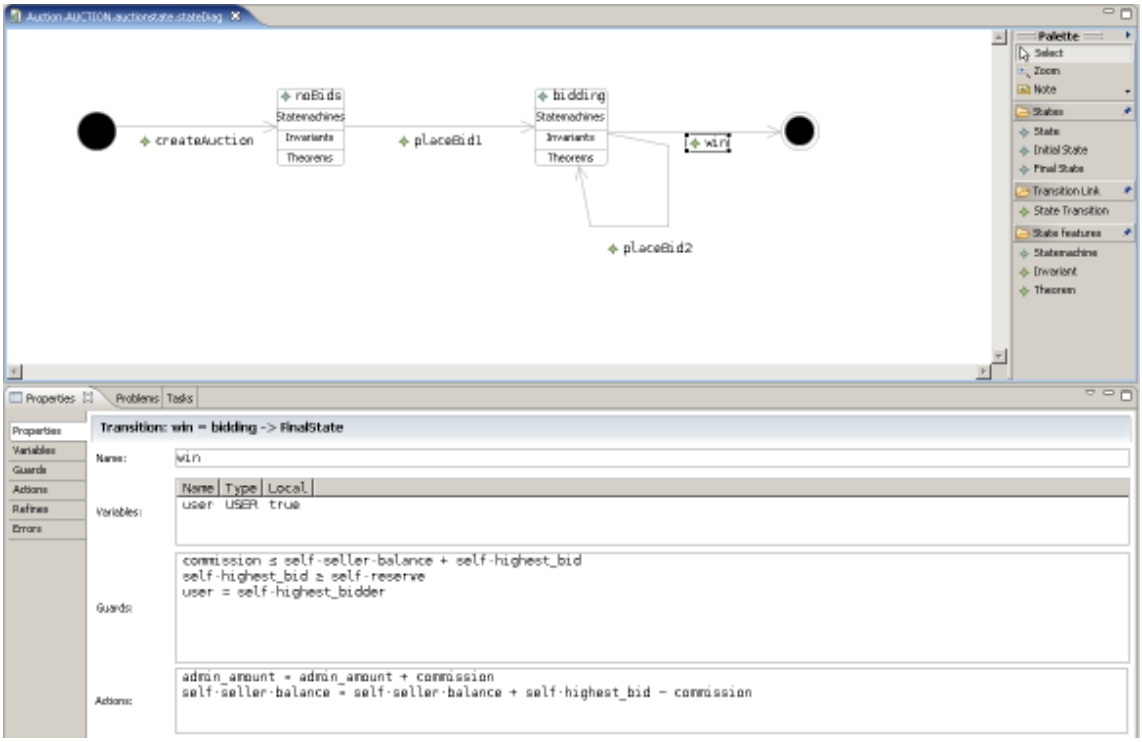


State Diagram for AUCTION Class – Continued



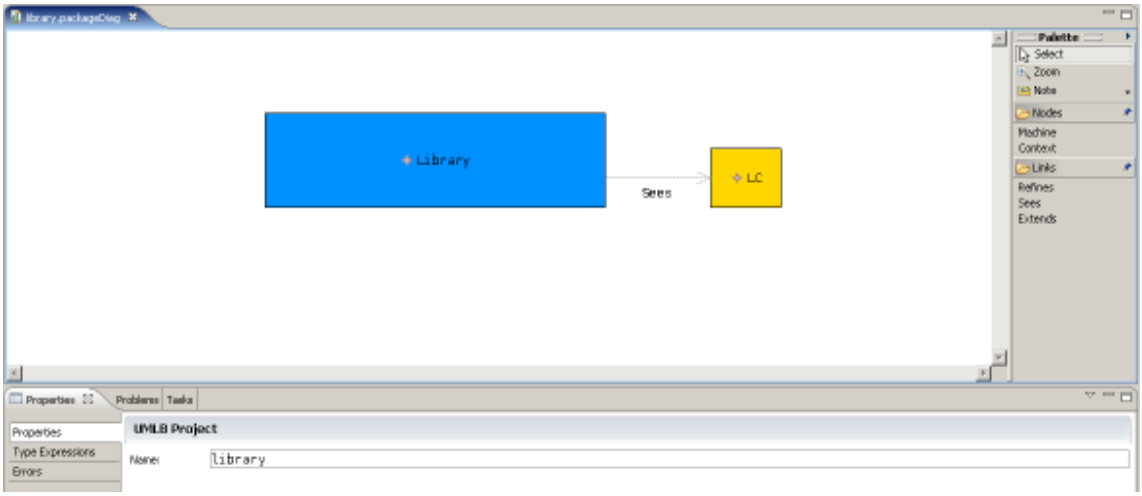


State Diagram for AUCTION Class – Continued

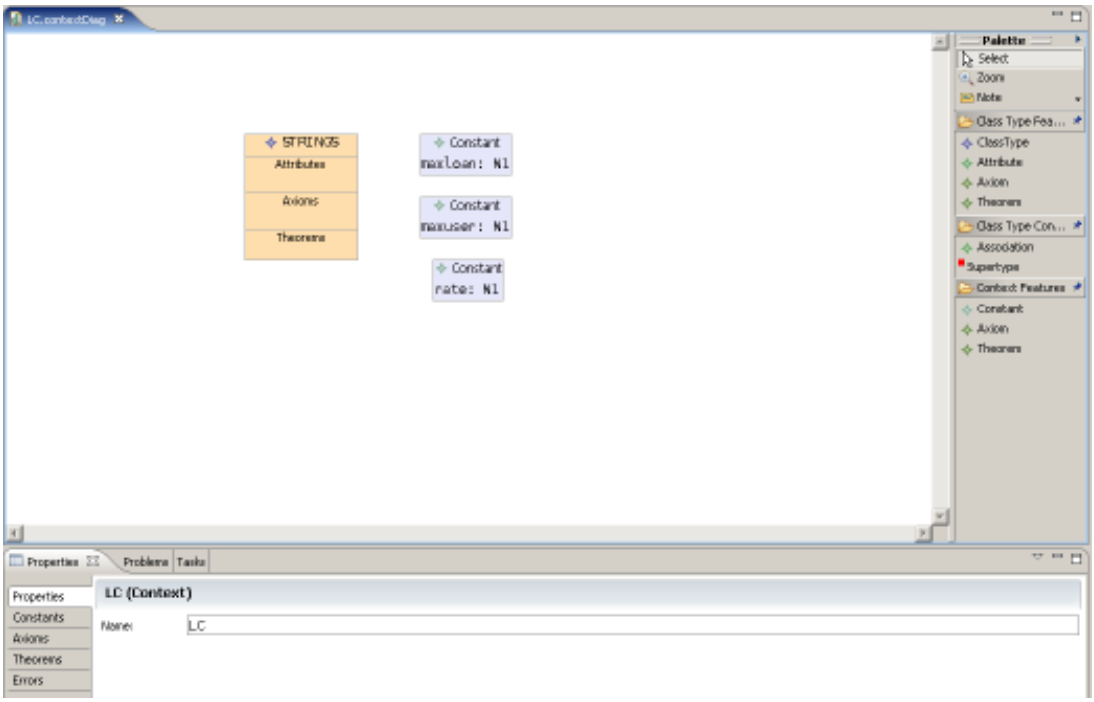


Case 2: Library System

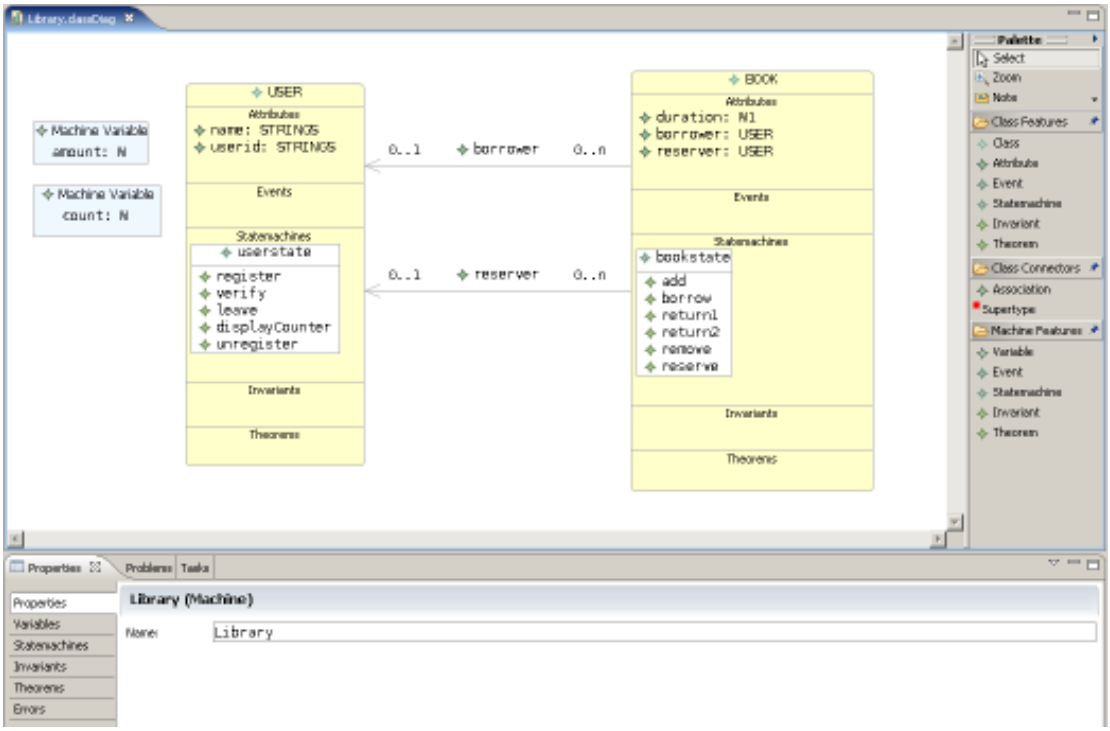
Package Diagram



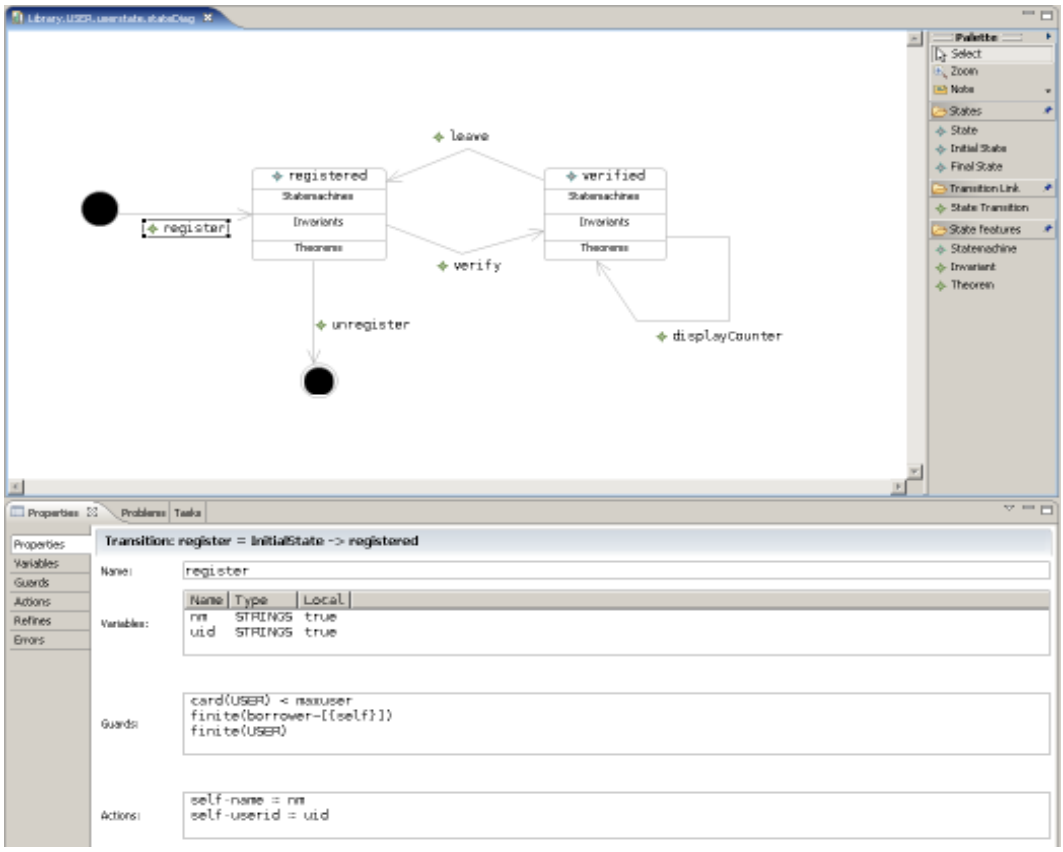
Context Diagram

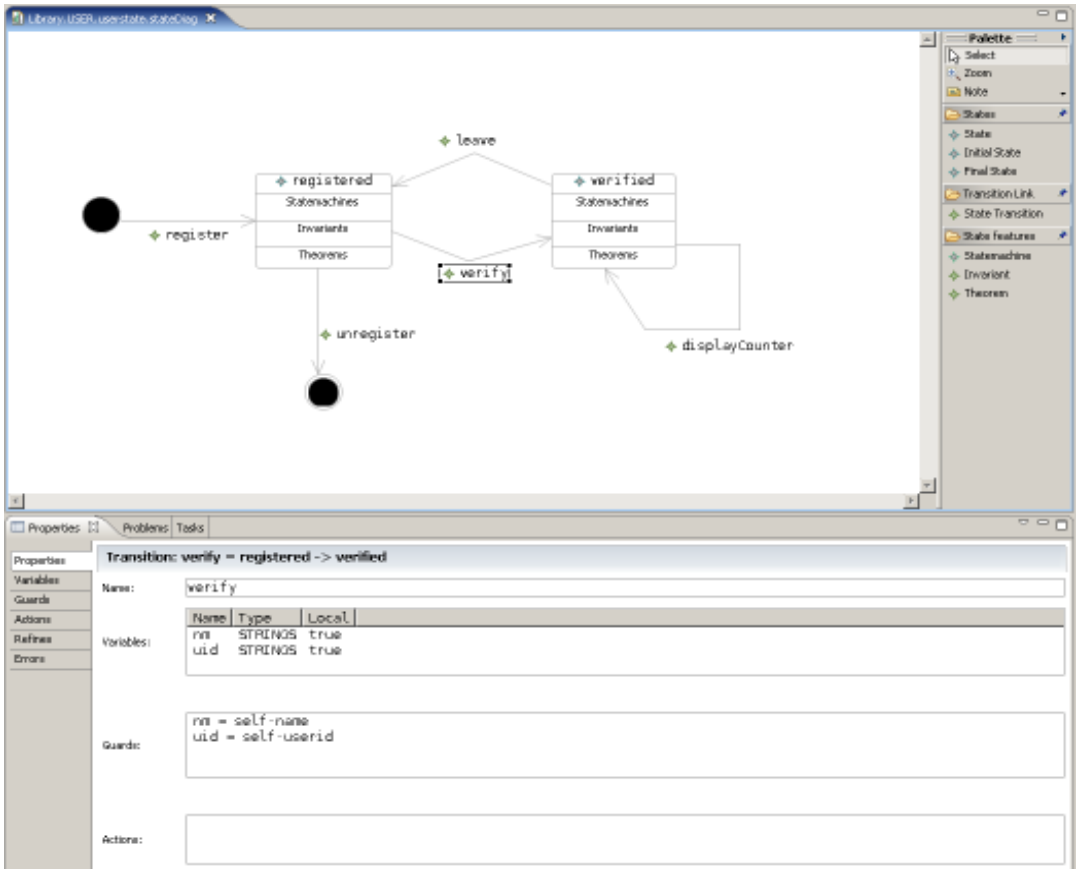


Class Diagram (Machines)

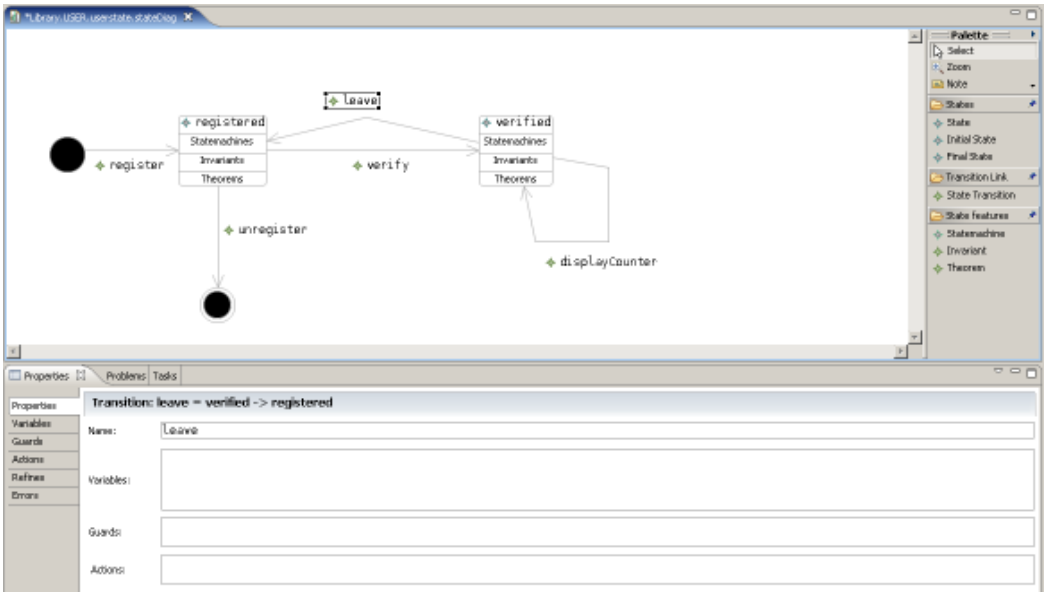


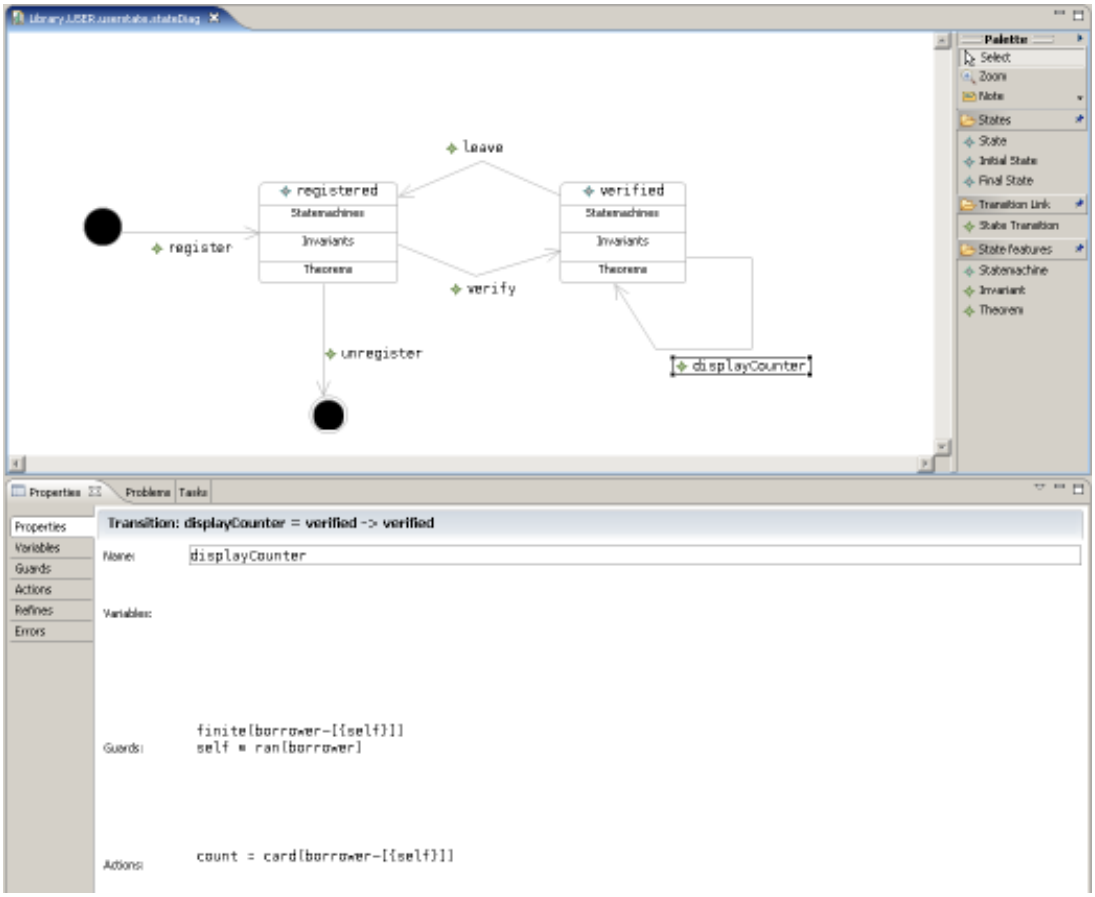
State Diagram for USER Class



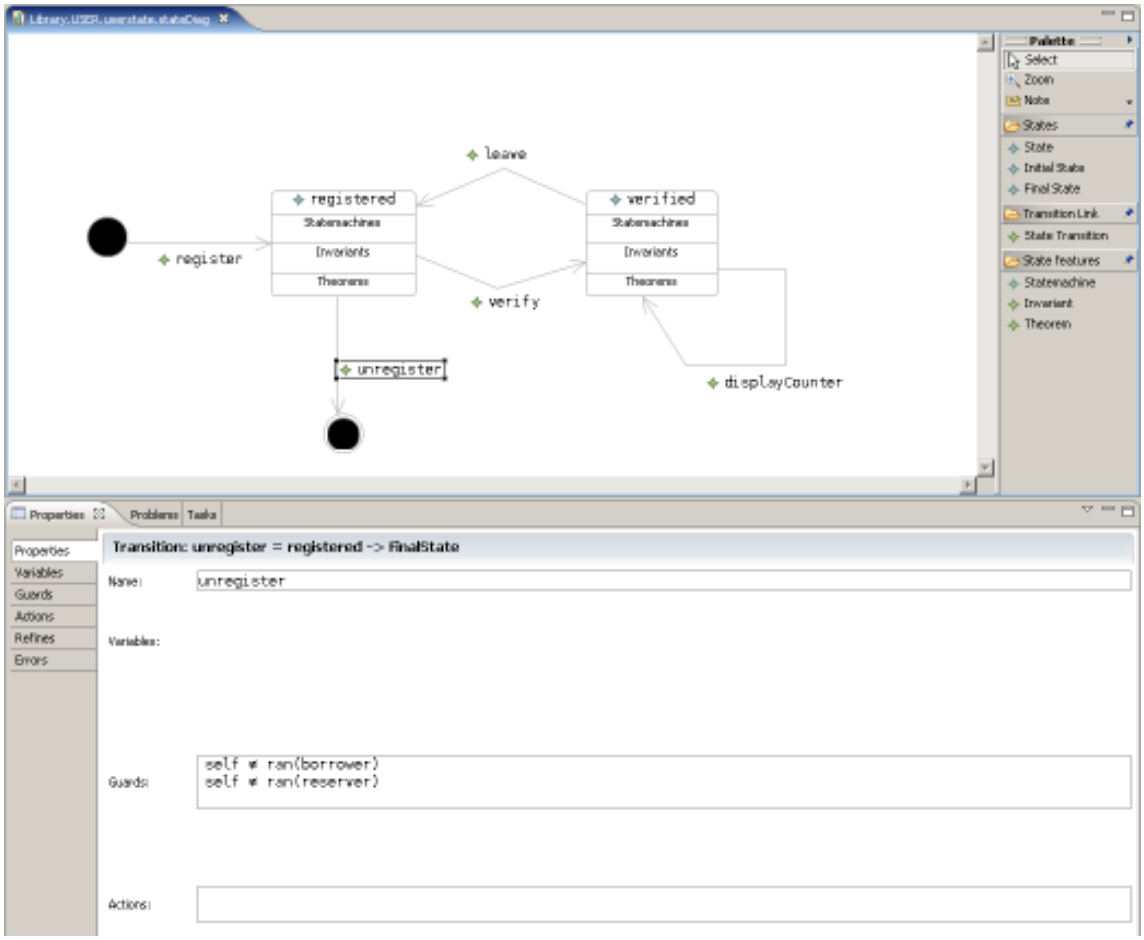


State Diagram for USER Class – Continued

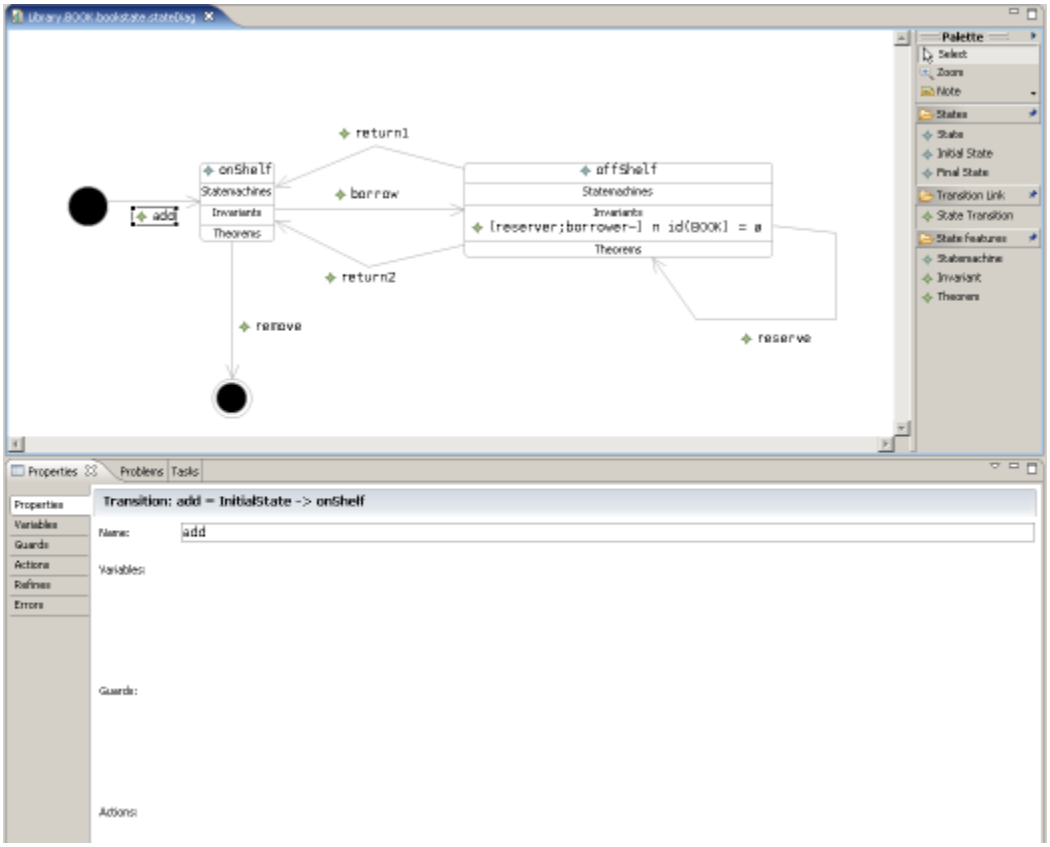


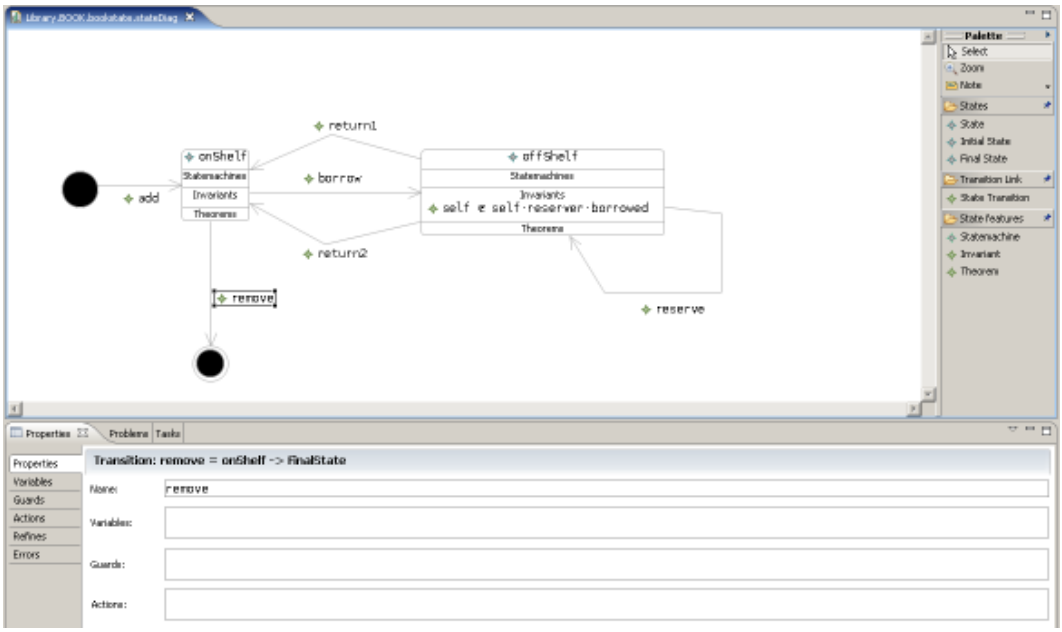


State Diagram for USER Class – Continued

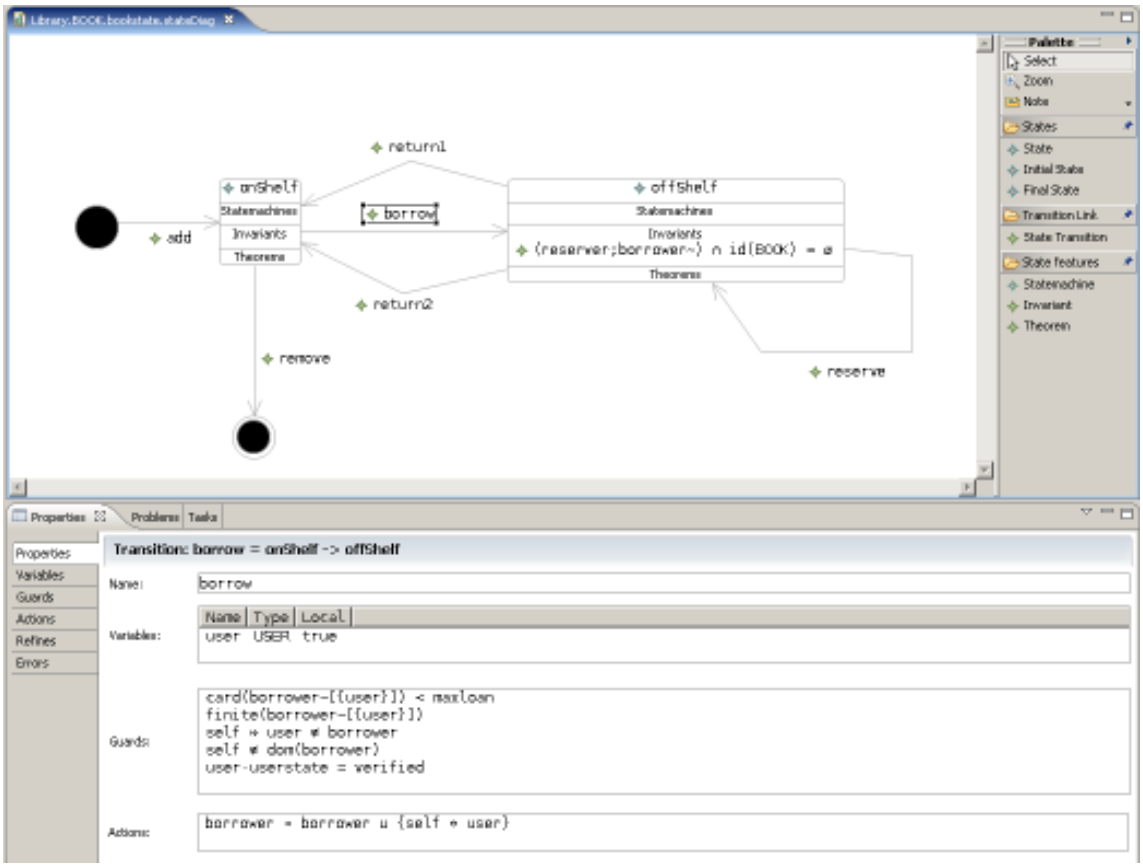


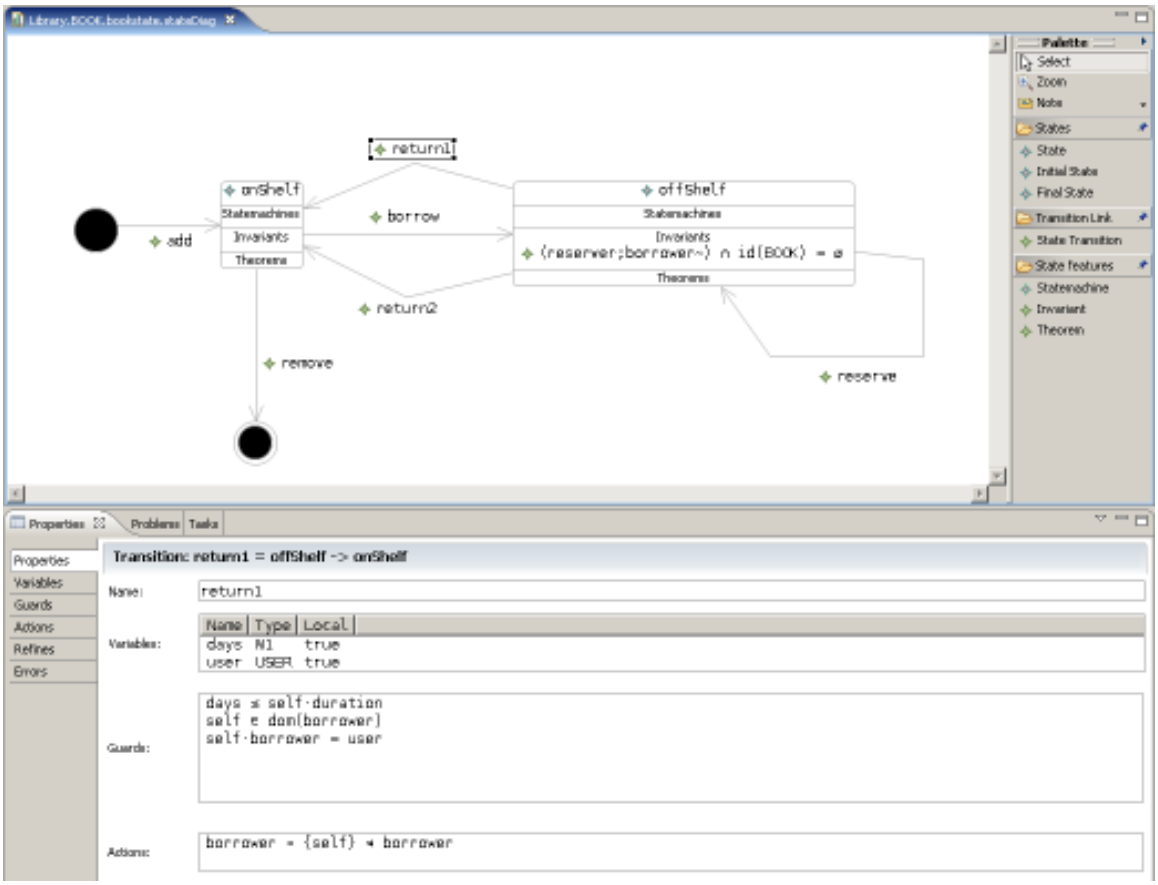
State Diagram for BOOK Class



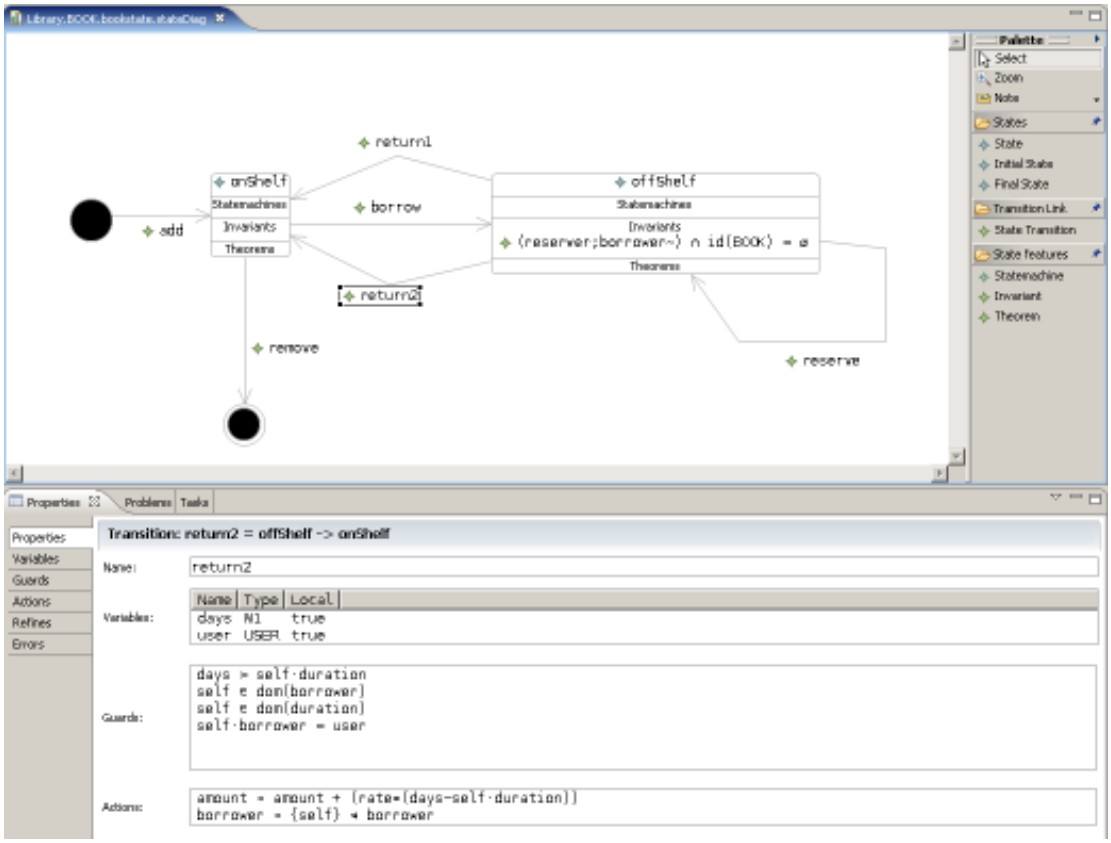


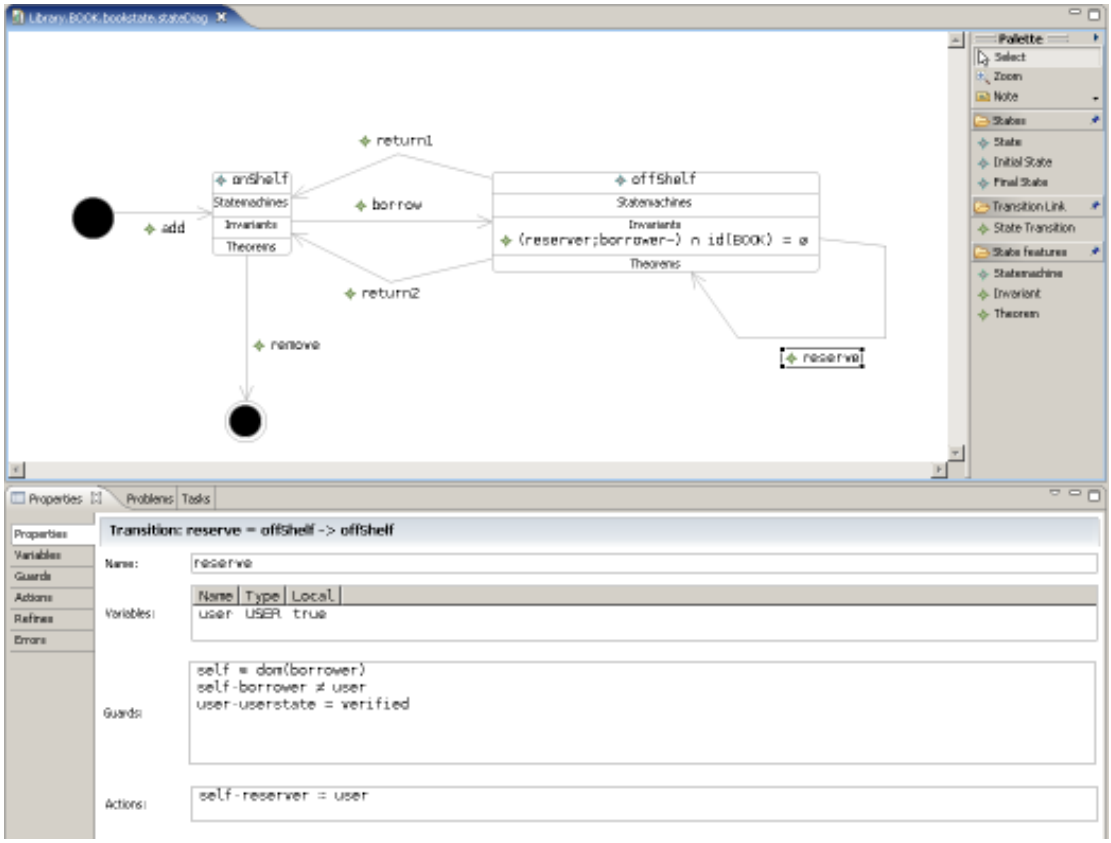
State Diagram for BOOK Class – Continued





State Diagram for BOOK Class – Continued





C.1.2. Event-B Models

Case 1: Auction System

Pretty Print

```
MACHINE Auction

SEES AC

VARIABLES
  auction
  registered
  reserve
  highest_bid
  seller
  highest_bidder
  balance
  name
  password
  admin_amount
  auction_state
  user_state
  amount

INVARIANTS
  inv1: auction ∈ ℙ(AUCTION)
  inv2: registered ∈ ℙ(USER)
  inv3: reserve ∈ auction → ℕ1
  inv4: highest_bid ∈ auction ⇸ ℕ
  inv5: seller ∈ auction → registered
  inv6: highest_bidder ∈ auction ⇸ registered
  inv7: balance ∈ registered → ℕ
  inv8: name ∈ registered ↣ STRINGS
  inv9: password ∈ registered → STRINGS
  inv10: auction_state ∈ auction → AUCTION_STATE
  inv11: user_state ∈ registered → USER_STATE
  inv12: admin_amount ∈ ℕ
  inv13: amount ∈ ℕ
  inv15: ∀aa·((aa ∈ auction)⇒(auction_state(aa) = noBids ∨ seller(aa) ≠ highest_bidder(aa)))
```

```
EVENTS

  INITIALISATION
    BEGIN
      act1: auction ≔ ø
      act2: registered ≔ ø
      act3: reserve ≔ ø
      act4: highest_bid ≔ ø
      act5: seller ≔ ø
      act6: highest_bidder ≔ ø
      act7: balance ≔ ø
      act8: name ≔ ø
      act9: password ≔ ø
      act10: admin_amount ≔ 0
      act11: auction_state ≔ ø
      act12: user_state ≔ ø
      act13: amount ≔ 0
    END

  createAuction
    ANY
      uu
      aa
      rr
    WHERE
      grd11: uu ∈ registered
      grd12: rr ∈ ℕ1
      grd13: aa ∈ AUCTION \ auction
      grd1: commission ≤ balance(uu)
      grd2: user_state(uu) = loggedIn
    THEN
      act11: auction ≔ auction ∪ {aa}
      act12: seller(aa) ≔ uu
      act13: reserve(aa) ≔ rr
      act1: balance(uu) ≔ balance(uu) − commission
      act2: admin_amount ≔ admin_amount + commission
      act3: highest_bid(aa) ≔ 0
      act4: auction_state(aa) ≔ noBids
    END
```

```
placeBid1
  ANY
    uu
    aa
    bb
  WHERE
    grd11: uu ∈ registered
    grd12: bb ∈ ℕ1
    grd13: aa ∈ auction
    grd1: aa ∉ dom(highest_bidder)
    grd2: uu ≠ seller(aa)
    grd3: bb ≥ reserve(aa)
    grd4: bb ≤ balance(uu)
    grd5: highest_bid(aa) = 0
    grd6: auction_state(aa) = noBids
    grd7: user_state(uu) = loggedIn
  THEN
    act11: highest_bid(aa) ≔ bb
    act12: highest_bidder(aa) ≔ uu
    act13: balance(uu) ≔ balance(uu) − bb
    act1: auction_state(aa) ≔ bidding
  END
placeBid2
  ANY
    uu
    aa
    bb
  WHERE
    grd11: uu ∈ registered
    grd12: aa ∈ auction
    grd13: bb ∈ ℕ1
    grd1: aa ∈ dom(highest_bidder)
    grd2: uu ≠ seller(aa)
    grd3: uu ≠ highest_bidder(aa)
    grd4: bb > highest_bid(aa)
    grd5: bb ≤ balance(uu)
    grd6: auction_state(aa) = bidding
    grd7: user_state(uu) = loggedIn
  THEN
    act11: highest_bidder(aa) ≔ uu
    act12: highest_bid(aa) ≔ bb
    act13: balance ≔ balance ∘ {uu ↦ balance(uu) − bb, highest_bidder(aa)
                    ↦ balance(highest_bidder(aa)) + highest_bid(aa)}
  END
```

```
win
  ANY
    uu
    aa
  WHERE
    grd11: uu ∈ registered
    grd12: aa ∈ auction
    grd13: uu = highest_bidder(aa)
    grd1: highest_bid(aa) ≥ reserve(aa)
    grd2: commission ≤ balance(seller(aa)) + highest_bid(aa)
    grd3: auction_state(aa) = bidding
  THEN
    act11: auction := auction \ {aa}
    act12: balance(seller(aa)) := balance(seller(aa)) + highest_bid(aa) − commission
    act13: admin_amount := admin_amount + commission
    act1: reserve := {aa} ⩤ reserve
    act2: highest_bid := {aa} ⩤ highest_bid
    act3: seller := {aa} ⩤ seller
    act4: highest_bidder := {aa} ⩤ highest_bidder
    act5: auction_state := {aa} ⩤ auction_state
  END

register
  ANY
    uu
    nn
    pp
  WHERE
    grd11: uu ∈ USER \ registered
    grd12: nn ∈ STRINGS
    grd13: pp ∈ STRINGS
    grd1: nn ∉ ran(name)
  THEN
    act11: registered := registered ∪ {uu}
    act12: name(uu) := nn
    act13: password(uu) := pp
    act1: balance(uu) := 0
    act2: user_state(uu) := loggedOut
  END

unregister
  ANY
    uu
  WHERE
    grd11: uu ∈ registered
    grd12: uu ∉ ran(seller)
    grd13: uu ∉ ran(highest_bidder)
    grd1: balance(uu) = 0
    grd2: user_state(uu) = loggedIn
  THEN
    act11: registered := registered \ {uu}
    act12: name := {uu} ⩤ name
    act13: password := {uu} ⩤ password
    act1: balance := {uu} ⩤ balance
    act2: user_state := {uu} ⩤ user_state
  END
```

```
  login
    ANY
      uu
      nn
      pp
    WHERE
      grd11: uu ∈ registered
      grd12: nn ∈ STRINGS
      grd13: pp ∈ STRINGS
      grd1: nn = name(uu)
      grd2: pp = password(uu)
      grd3: user_state(uu) = loggedOut
    THEN
      act11: user_state(uu) := loggedIn
    END

  logoff
    ANY
      uu
    WHERE
      grd11: uu ∈ registered
      grd12: user_state(uu) = loggedIn
    THEN
      act11: user_state(uu) := loggedOut
    END

  displayAmount
    ANY
      uu
    WHERE
      grd11: uu ∈ registered
      grd12: user_state(uu) = loggedIn
    THEN
      act11: amount := balance(uu)
    END

  displayAdminAmount
    BEGIN
      act11: amount := admin_amount
    END

END
```

Case 2: Library System

Pretty Print

```
MACHINE Library

SEES LC

VARIABLES
  book
  user
  duration
  reserver
  borrower
  name
  userid
  amount
  book_state
  user_state
  count

INVARIANTS
  inv1: book ∈ ℙ(BOOK)
  inv2: user ∈ ℙ(USER)
  inv3: duration ∈ book → ℕ1
  inv4: reserver ∈ book ⇸ user
  inv5: borrower ∈ book ⇸ user
  inv6: name ∈ user → STRINGS
  inv7: userid ∈ user ↣ STRINGS
  inv8: amount ∈ ℕ
  inv9: book_state ∈ book → BOOK_STATE
  inv10: user_state ∈ user → USER_STATE
  inv11: (reserver;borrower~) ∩ id(book) = ∅
  inv12: ∀uu·(uu ∈ user ∧ finite(borrower~[{uu}]) ⇒ card(borrower~[{uu}]) ≤ maxloan)
  inv13: count ∈ ℕ
  inv14: finite(user) ∧ card(user) ≤ maxuser
```

```
EVENTS

  INITIALISATION
    BEGIN
      act1: book ≔ ∅
      act2: user ≔ ∅
      act3: duration ≔ ∅
      act4: reserver ≔ ∅
      act5: borrower ≔ ∅
      act6: name ≔ ∅
      act7: userid ≔ ∅
      act8: amount ≔ 0
      act9: count ≔ 0
      act10: book_state ≔ ∅
      act11: user_state ≔ ∅
    END

  add
    ANY
      bb
    WHERE
      grd11: bb ∈ BOOK ∖ book
    THEN
      act11: book ≔ book ∪ {bb}
      act12: book_state(bb) ≔ onShelf
      act13: duration(bb) ≔ 30
    END

  remove
    ANY
      bb
    WHERE
      grd11: bb ∈ book
      grd12: book_state(bb) = onShelf
    THEN
      act11: book ≔ book ∖ {bb}
      act12: book_state ≔ {bb} ⩤ book_state
      act13: duration ≔ {bb} ⩤ duration
      act1: reserver ≔ {bb} ⩤ reserver
      act2: borrower ≔ {bb} ⩤ borrower
    END
```

```
borrow
  ANY
    bb
    uu
  WHERE
    grd11: bb ∈ book
    grd12: uu ∈ user
    grd13: user_state(uu) = verified
    grd2: bb ∉ dom(borrower)
    grd6: card(borrower~[{uu}]) < maxloan
    grd4: book_state(bb) = onShelf
    grd7: bb ↦ uu ∉ borrower
    grd1: finite(borrower~[{uu}])
  THEN
    act11: book_state(bb) := offShelf
    act12: borrower := borrower ∪ {bb ↦ uu}
  END

return1
  ANY
    bb
    uu
    dd
  WHERE
    grd11: bb ∈ book
    grd12: uu ∈ user
    grd13: dd ∈ ℕ1
    grd1: bb ∈ dom(borrower)
    grd4: borrower(bb) = uu
    grd2: dd ≤ duration(bb)
    grd3: book_state(bb) = offShelf
  THEN
    act11: book_state(bb) := onShelf
    act12: borrower := {bb} ⩤ borrower
  END

return2
  ANY
    bb
    uu
    dd
  WHERE
    grd11: bb ∈ book
    grd12: uu ∈ user
    grd13: dd ∈ ℕ1
    grd1: bb ∈ dom(borrower)
    grd4: borrower(bb) = uu
    grd5: bb ∈ dom(duration)
    grd2: dd > duration(bb)
    grd3: book_state(bb) = offShelf
  THEN
    act11: book_state(bb) := onShelf
    act12: borrower := {bb} ⩤ borrower
    act13: amount := amount + (rate * (dd-duration(bb)))
  END
```

```
reserve
  ANY
    bb
    uu
  WHERE
    grd11: bb ∈ book
    grd12: uu ∈ user
    grd13: user_state(uu) = verified
    grd1: bb ∈ dom(borrower)
    grd3: book_state(bb) = offShelf
    grd2: borrower(bb) ≠ uu
  THEN
    act11: reserver(bb) ≔ uu
  END

register
  ANY
    uu
    nn
    ii
  WHERE
    grd11: uu ∈ USER \ user
    grd12: nn ∈ STRINGS
    grd13: ii ∈ STRINGS \ ran(userid)
    grd2: finite(user)
    grd1: card(user) < maxuser
    grd3: finite(borrower~[{uu}])
  THEN
    act11: user ≔ user ∪ {uu}
    act12: user_state(uu) ≔ registered
    act13: name(uu) ≔ nn
    act1: userid(uu) ≔ ii
  END

unregister
  ANY
    uu
  WHERE
    grd11: uu ∈ user
    grd12: user_state(uu) = registered
    grd13: uu ∉ ran(reserver)
    grd1: uu ∉ ran(borrower)
  THEN
    act11: user ≔ user \ {uu}
    act12: user_state ≔ {uu} ⩤ user_state
    act13: name ≔ {uu} ⩤ name
    act1: userid ≔ {uu} ⩤ userid
  END
```

```
    verify
      ANY
        uu
        nn
        ii
      WHERE
        grd11: uu ∈ user
        grd12: nn ∈ STRINGS
        grd13: ii ∈ STRINGS
        grd1: nn = name(uu)
        grd2: ii = userid(uu)
        grd3: user_state(uu) = registered
      THEN
        act11: user_state(uu) ≔ verified
      END

    leave
      ANY
        uu
      WHERE
        grd11: uu ∈ user
        grd12: user_state(uu) = verified
      THEN
        act11: user_state(uu) ≔ registered
      END

    displayCounter
      ANY
        uu
      WHERE
        grd11: uu ∈ user
        grd12: user_state(uu) = verified
        grd1: uu ∈ ran(borrower)
        grd2: finite(borrower~[{uu}])
      THEN
        act11: count ≔ card(borrower~[{uu}])
      END

    displayAmount
      BEGIN
        act11: count ≔ amount
      END

END
```

C.2. Questions

C.2.1. UML-B and Event-B Models Questionnaires

Case 1: Auction System

Question 1

Describe briefly in **natural language** the main ideas illustrated in the given model (i.e. a high level description/summary of the **key** entities involved and the relationships/interactions between them).

Question 2

Consider the following situation:

Fred has created an auction in the system. Although the current bid on his auction is high, he is eager for gaining more. He plans to be a "catalyst" by placing a higher bid than the current one. He hopes the current bidder or other users will outbid him later.

How does the given model handle this sort of situation? Explain.

Note: "Catalyst" – medium for a change

Question 3

A new requirement is added to the system:

A user can change his/her password. He/she needs to provide his/her current password and propose a new password. The new password must be different from the current one. The number of times the password has been changed must not exceed the maximum number of times allowed. He/she must log in for this operation.

How would you enhance the given UML-B/Event-B model to include this new requirement?

Formulate your solution explicitly by sketching the necessary elements on the given **Answer Sheets for Question 3**. State the semantics (i.e. attributes, event name, variables, guards and actions etc.) on the respective parts of the screenshots.

Reminder: You should state your answers on the given sheets. **Do not** do any modification on the model online (using computer).

Question 4

There are two different situations/restrictions of placing a bid in the given model. Compare and contrast them by clearly explaining the differences.

Question 5

Eventually, each created auction must be closed. What are the conditions for this to happen? What actions should be taken by the system? Propose an idea (plan) in **natural language** for meeting this requirement.

Question 6

Criticise the given model. Suggest ways (as much as you could think) that the model should be improved to represent a better **Auction** system.

Case 2: Library System

Question 1

Describe briefly in **natural language** the main ideas illustrated in the given model (i.e. a high level description/summary of the **key** entities involved and the relationships/interactions between them).

Question 2

Consider the following situation:

Sam brings the "Chemistry I" book that he borrowed last week to the library. Before returning the book to the counter, he goes to the shelf to find a book, titled "Chemistry II". While he is looking for the book, he has accidentally left the "Chemistry I" book on the shelf where he begins the searching. He forgets about it and then leaves the library. Later, Cindy finds the "Chemistry I" book that Sam left. She feels like borrowing it.

How does the given model handle this sort of situation? Explain.

Question 3

A new requirement is added to the system:

A user can renew a book provided another user has not reserved the book. The number of times the book has been renewed by him/her must not exceed the maximum number of renewals allowed. His/her identification must be verified for this operation.

How would you enhance the given UML-B/Event-B model to include this new requirement?

Formulate your solution explicitly by sketching the necessary elements on the given **Answer Sheets for Question 3**. State the semantics (i.e. attributes, event name, variables, guards and actions etc.) on the respective parts of the screenshots.

Reminder: You should state your answers on the given sheets. **Do not** do any modification on the model online (using computer).

Question 4

There are two different situations/restrictions of returning a book in the given model. Compare and contrast them by clearly explaining the differences.

Question 5

What if more than one user wants to reserve the same book? And, what should happen later when the current borrower returns the reserved book? Propose an idea (plan) in **natural language** indicating the conditions and actions that should be taken by the system for meeting this requirement.

Question 6

Criticise the given model. Suggest ways (as much as you could think) that the model should be improved to represent a better **Library** system.

C.2.2 Debriefing Questionnaire

Problem Strategies

Which strategy did you use to answer the question? Please **tick one or more** that best describe your strategy.

- □ Heuristics/Guess/Hunch
- □ Used a specific past experience (a known solution of a similar problem) and applied it by analogy (comparison)
- □ Used an existing definite solution that satisfies this problem completely
- □ Used a general (logical) solution and then refined it to suit this problem
- □ Other, please specify.

Comprehension Strategies

Read the following strategies.

Strategy A
I began by firstly making a general assumption about the system described by the model based on the information available at first glance (i.e. class titles, associations, model structure). The assumption led me to expect certain attributes and events/operations in each class, which resulted in another level or more specific assumptions. I later continued looking for specific information in events/operations to verify (i.e. accept or reject) the assumptions.

Strategy B
I began by firstly analysing a specific event/operation of a class. I understood the functionality of that event/operation locally. That individual event/operation was later linked with other relevant parts in the model to understand the relationships between events/operations and their roles in a specific class. In this way, I obtained a unit of information about the model. Later, I repeated the process until the whole model was reviewed and understood.

In general, which strategy did you use to comprehend the **whole** model? Please **tick one** that best describes your strategy.

- □ **Strategy A**, or
- □ **Strategy B**, or
- □ **Other**, please specify.

Model Comprehensibility

Please answer the following questions as sincerely as possible. Your answers will be treated confidentially.

Having done some exercises on both models:

How would you rate UML-B model comprehensibility? Circle the answer.

Very difficult to comprehend						**Very easy to comprehend**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Which parts of a UML-B model that could ease understanding?

Which parts of a UML-B model that could hinder understanding?

How would you rate Event-B model comprehensibility? Circle the answer.

Very difficult to comprehend						**Very easy to comprehend**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Which parts of an Event-B model that could ease understanding?

Which parts of an Event-B model that could hinder understanding?

Model Preference

If you are given the choice, which model would you prefer to deal with: **UML-B** or **Event-B** (or classical B)? Why?

Comments on the Models

Any other comments (i.e. positive and/or negative feedback)?

C.3. Raw Data

Case 1: Auction System

Subject	Question 1	Question 2	Question 3	Question 4	Question 5	Question 6
Case 1: UML-B (Unit: Marks/min)						
A1	1.12	0.67	2.38	1.14	0.67	0.00
A2	1.14	0.30	0.29	0.25	0.25	0.33
A3	0.00	0.60	0.00	1.60	0.40	0.20
A4	1.07	0.44	1.50	0.50	N/A	N/A
A5	0.60	0.40	0.91	1.00	0.10	N/A
A6	1.50	0.19	1.32	0.57	N/A	N/A
A7	0.71	0.67	2.83	0.77	0.25	1.00
A8	1.13	1.00	5.75	2.00	0.50	1.00
A9	0.53	0.50	0.20	0.14	0.00	0.00
A10	1.00	0.00	0.90	1.00	0.50	N/A
A11	0.33	0.00	0.19	1.00	0.33	0.40
A12	1.29	0.07	0.23	N/A	N/A	N/A
A13	0.39	0.38	1.60	0.80	0.33	N/A
A14	0.53	0.20	0.00	N/A	N/A	N/A
A15	1.10	0.75	2.17	1.00	1.00	0.60
A16	0.31	0.50	0.85	0.00	1.25	N/A
A17	0.83	0.40	1.42	0.50	0.60	N/A
A18	0.50	0.50	0.67	0.78	0.67	1.00
Case 1: Event-B (Unit: Marks/min)						
B1	0.75	1.00	1.50	1.25	0.67	0.00
B2	1.22	0.13	N/A	0.00	1.00	0.00
B3	0.45	0.60	N/A	0.67	0.17	N/A
B4	0.53	0.67	1.71	1.60	0.29	0.33
B5	0.31	0.11	1.38	0.40	0.50	0.00
B6	0.73	0.50	0.91	0.86	0.33	0.00
B7	0.36	0.00	0.60	0.80	0.17	N/A

B8	0.50	0.00	0.50	0.86	0.00	N/A
B9	0.36	0.50	1.17	0.00	N/A	N/A
B10	N/A	N/A	N/A	N/A	N/A	N/A
B11	0.45	0.00	1.70	1.00	0.33	N/A
B12	0.45	0.50	N/A	0.47	0.00	0.00
B13	1.00	0.67	1.67	0.00	1.00	0.20
B14	0.50	0.67	0.69	2.00	N/A	N/A
B15	0.90	2.00	2.25	0.67	1.00	N/A
B16	0.77	0.67	0.50	0.67	1.00	N/A
B17	0.00	0.25	0.14	1.00	0.40	0.00
B18	0.73	0.33	1.06	1.20	0.75	0.67

Case 2: Library System

Subject	Question 1	Question 2	Question 3	Question 4	Question 5	Question 6
Case 2: UML-B (Unit: Marks/min)						
B1	1.75	0.67	1.50	3.50	1.33	N/A
B2	2.25	0.17	1.30	2.50	1.33	0.50
B3	0.60	0.25	N/A	1.17	0.60	0.14
B4	1.71	1.00	2.13	0.80	0.25	0.50
B5	1.63	0.50	2.00	0.86	1.00	1.50
B6	0.83	0.38	0.65	1.17	N/A	N/A
B7	1.44	0.17	1.90	0.86	2.00	1.00
B8	2.00	0.00	0.50	2.00	N/A	N/A
B9	2.92	0.25	2.00	2.00	1.00	0.00
B10	N/A	N/A	N/A	N/A	N/A	N/A
B11	1.00	0.60	0.85	0.30	0.60	1.00
B12	0.69	0.50	N/A	0.80	1.00	0.23
B13	1.30	1.00	2.14	6.00	1.00	0.33
B14	0.78	0.40	0.70	1.67	0.57	0.80
B15	2.00	0.25	2.80	2.00	1.00	N/A
B16	2.00	0.33	1.56	2.50	1.25	0.25
B17	1.83	1.00	1.28	1.00	0.33	N/A
B18	1.75	0.50	0.80	1.50	0.50	1.00
Case 2: Event-B (Unit: Marks/min)						
A1	1.20	0.13	0.88	1.50	1.25	1.00
A2	1.50	0.00	N/A	0.00	0.20	0.67
A3	0.71	0.67	N/A	1.17	0.40	0.08
A4	0.00	0.17	0.68	3.50	0.00	0.00
A5	0.25	0.67	0.94	5.00	0.40	N/A
A6	1.00	0.50	1.57	0.33	1.50	N/A
A7	1.80	0.29	1.31	1.25	1.00	0.00
A8	0.89	0.29	1.20	1.67	0.40	0.00
A9	1.00	0.00	1.38	5.00	1.00	0.00
A10	0.75	0.17	0.43	0.83	0.67	0.13
A11	0.36	0.20	0.20	0.83	1.00	0.40
A12	0.27	0.07	0.17	1.67	0.50	N/A
A13	0.83	0.33	1.04	1.00	1.00	0.00
A14	0.69	0.15	0.73	N/A	N/A	N/A
A15	1.30	0.00	1.11	1.75	1.25	0.50
A16	1.33	0.75	0.77	2.50	0.67	0.00
A17	1.29	0.60	0.73	0.86	0.67	0.00
A18	0.40	0.40	0.88	1.25	1.25	0.25

Appendix D

Survey 2

This appendix presents the questions used in the second survey (**Chapter 6**).

Questions:

Please **underline** (or any suitable indicator) the option that most accurately describes your answer and please provide the necessary information at the end of each question.

(1) If you need to compare different parts of your **UML-B model** (e.g. between class and statechart diagrams, machines and contexts, properties of different events, switching navigation panes etc.), how easy is it to view them at the same time in **Rodin Tool**?

Very difficult						**Very easy**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Why?

(2) If you need to rebuild/restructure your **UML-B model** (e.g. due to change in ideas or requirements or solutions), how easy is it to make the changes?

Note: The changes include editing the diagrams and the respective semantics.

Very difficult						**Very easy**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Are there any particular changes that are particularly difficult or tedious to make?

No **Not Sure** **Yes**

If **Yes**, which ones?

(3) How simple (straightforward) is it to describe what you intend when modeling your **UML-B model**?

Very complicated (long winded)						**Very easy (reasonably brief)**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Why?

(4)

(i) How easy is it to make mistakes when modeling the **diagrams** in your **UML-B model**?

Very difficult						**Very easy**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Why?

(ii) How easy is it to make mistakes when defining the formal semantics in **microB clauses** for the diagrams in your **UML-B model**?

Very difficult						**Very easy**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Why?

(5) Can you stop modeling your **UML-B model** (partially completed) at any time you like and check your work so far?

No **Not Sure** **Yes**

Why?

(6) Do you find any complex or difficult tasks to work out in your head when modeling your **UML-B model**?

No **Not Sure** **Yes**

If **Yes**, what are they?

(7) Are there any parts in the **UML-B model** that seem to be similar in functionality (i.e. mean or operate similar things) but the **UML-B method** makes them appear different?

No **Not Sure** **Yes**

If **Yes**, what are they?

(8) Do you find any structure dependencies in **UML-B model** (i.e. one part explicitly relies upon or is determined by or uses or requires another part)?

No **Not Sure** **Yes**

If **Yes**, how visible are the structure dependencies?

(a) Visible in both parts
(b) Visible in one part
(c) Not visible in both parts

If **(b)** or **(c)**, what are the parts involved? Please state them specifically.

(9) Does **Rodin Tool** allow you to make notes or convey extra information beyond the **UML-B model** to yourself (e.g. comments, use different fonts, layout)?

No **Not Sure** **Yes**

If **Yes**, please state the possible actions.

(10)

(i) How easy is it to determine what each **diagram** (and its components) is for in the **UML-B model** as a whole?

Very difficult						**Very easy**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Why?

(ii) How easy is it to determine what each **microB clause** is for in the **UML-B model** as a whole?

Very difficult						**Very easy**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Why?

(iii) Are there any parts that you simply include just because it is always been that way (without exactly knowing what the purposes)?

No **Not Sure** **Yes**

If **Yes**, what are they?

(11) How well does the **UML-B method** allow you to describe your problem accurately and completely as what you intend?

Very bad						**Very good**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Why?

(12) How well does the **UML-B method** allow you to play around with your model (e.g. when you are testing your ideas/solutions, without being sure what the effects will be)?

Very bad						**Very good**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Which part of the method (i.e. concepts, certain functions in the supporting tools etc.) help or prevent you to do this? State explicitly.

(13) When you are working with your model, does the **UML-B method** :

(a) allow you to go about any task in any order , OR
(b) enforce you to think ahead and make certain decisions first?

If **(a)**, please give some examples.
If **(b)**, what decisions do you need to make in advance?

(14) Does the **UML-B method** insist you start the modeling task by defining or grouping things first before you can do anything else?

No **Not Sure** **Yes**

If **Yes**, what sort of things?

(15) How easy is it to learn the **UML-B method** compared to the traditional **B method** and **Event-B method**?

Very difficult						**Very easy**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Are there any particular parts in the **UML-B method** that are particularly difficult to learn and understand how they work?

No	**Not Sure**	**Yes**

If **Yes**, which ones?

(16) How useful do you find the available manual and documentation (including online help) on the **UML-B method**?

Very useless						**Very useful**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Why?

(17) How easy is it to become familiar with the **UML-B method** and be able to use it in your task efficiently without referring to the documentation/online help?

Very difficult						**Very easy**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Why?

(18) How easy is it to do modeling using the **UML-B method** compared to the traditional **B method** and **Event-B method**?

Very difficult						**Very easy**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

If you are given the choice in modelling, which method would you choose:

UML-B	**B**	**Event-B**

Why?

(19) Do you find yourself using the **UML-B method** in ways that are unusual or ways that the method designer might not have intended? Can you think of obvious ways that the design of the **UML-B method** could be improved? What are they?

Appendix E

Survey 3

This appendix presents the questions used in the third survey (**Chapter 7**). This includes ProB and B-Toolkit questionnaires.

E.1. ProB Questionnaires

Questions:

Please **underline** (or any suitable indicator) the option that most accurately describes your answer and please provide the necessary information at the end of each question.

(1) How easy is it to view and search the various features in ProB when you are working with your B model?

Very difficult						**Very easy**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Why?

(2) If you need to rebuild/restructure your B model in (e.g. due to change in ideas or requirements), how easy is it to make the changes in ProB?

Note: The changes include the model editing in the editor pane, reloading the edited model, model re-verification and re-validation using the verification and animation features.

Very difficult						**Very easy**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Why?

(3) Does ProB let you do what you want to your B model reasonably straightforward?

No **Not Sure** **Yes**

If **No**, what sorts of things take more time and effort to accomplish?

(4)

(i) If you need to compare different parts of a B model (i.e. a machine), how easy is it to view several parts of the model at the same in ProB?

Very difficult						**Very easy**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Why?

(ii) If you need to compare different B models (i.e. more than one machine), how easy is it to view several B models at the same time in ProB?

Very difficult						**Very easy**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Why?

(5)

(i) If you verify your B model in ProB, how difficult is it to comprehend what is happening?

Very difficult						**Very easy**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Why?

(ii) If you animate your B model in ProB, how difficult is it to comprehend what is happening?

Very difficult						**Very easy**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Why?

(6) How easy is it to recognise and interpret the roles of available features in ProB?

Very difficult						**Very easy**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Are there any features that are particularly difficult to interpret (i.e. you are not sure what they mean and their purposes)? Which ones?

(7) How well does ProB allow you to play around with your B model (e.g. when you are testing your ideas/solutions, without being sure what the effects will be)?

Very bad						**Very good**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

What features of the tool help or prevent you to do this?

(8) Do some kinds of mistake seem particularly common or easy to make (i.e. do you often find yourself making small slips that irritate you or make you feel foolish)?

No **Not Sure** **Yes**

If **Yes**, which ones?

(9) Can you animate or verify your partially completed B model in ProB?

No **Not Sure** **Yes**

Why?

(10)

(i) Can you go about any task in any order you like in ProB?

No **Not Sure** **Yes**

Why?

(ii) Does ProB enforce you to think ahead and make certain decisions first before you could use it?

No **Not Sure** **Yes**

If **Yes**, what decisions do you need to make in advance?

(11) Do you find any features in ProB seem to be a strange way of working with a B model (or B method)?

No **Not Sure** **Yes**

If **Yes**, what are they?

(12) Do you find any feature dependencies in ProB (i.e. one feature explicitly relies upon or is determined by or uses or requires another feature)?

No **Not Sure** **Yes**

If **Yes**, how visible are the feature dependencies?

(a) Visible in both features
(b) Visible in one feature
(c) Not visible in both features

If **(b)** or **(c)**, what are the features involved? Please state them specifically.

(13) Does ProB allow you to make notes or convey extra information beyond the B model to yourself in the editor pane (e.g. use comments, different fonts, colour)?

No **Not Sure** **Yes**

If **Yes**, please state the possible actions.

(14) Does ProB insist you start the modeling task by defining or grouping things before you can do anything else?

No **Not Sure** **Yes**

If **Yes**, what sorts of things?

(15) Are there any features that seem to be similar in functionality (i.e. mean or operate similar things) but ProB makes them appear different?

No **Not Sure** **Yes**

If **Yes**, what are they?

(16) How easy is it to become familiar with ProB features and be able to use it in your task efficiently without referring to manuals/documentation?

Very difficult						**Very easy**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Why?

(17) Has ProB helped you to find problems in your B models (i.e. are the error messages during analysis/verification easy to understand and helpful, is the animation useful etc.)?

No **Not Sure** **Yes, a little** **Yes** **Yes, a lot**

Why?

(18) How useful do you find the online documentation in ProB (e.g. **About** menu)?

Very useless						**Very useful**
-3	**-2**	**-1**	**0**	**1**	**2**	**3**

Why?

(19) Has ProB helped you to understand or learn the B method?

No **Not Sure** **Yes, a little** **Yes** **Yes, a lot**

Why?

(20)

(i) Which features do you use most often (e.g. Temporal Model Checking, Visualisation of State Space, Analyse Invariant etc.)?

(ii) Can you think of obvious ways that the design of the tool could be improved? Which new feature(s) or command(s) would be most helpful?

E.2. B-Toolkit Questionnaires

Questions:

Please circle the option that most accurately describes your answer.

(1) How easy is it to view and search the various features in the B-Toolkit when you are working with your B model?

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

Why?

(2) If you need to rebuild/restructure your B model (e.g. due to change in ideas or requirements to your previous model), how easy is it to make the changes in the B-Toolkit?

Note: The changes include edit, re-analyse, re-animate and re-verify the B model.

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

Why?

(3) Does the B-Toolkit let you do what you want to your B model reasonably straightforward?

Yes **No** **Not Sure**

If **No**, what sorts of things take more time and effort to accomplish?

(4) If you need to compare different parts of a B model (i.e. a machine), how easy is it to view several parts of a model at the same time in the B-Toolkit?

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

Why?

If you need to compare different B models (i.e. more than one machine), how easy is it to view several B models at the same time in the B-Toolkit?

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

Why?

(5) If you analyse your B model in the B-Toolkit using *Analyser* (i.e. *anl*), how easy is it to comprehend what is happening?

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

What kind of things requires the most mental effort?

If you animate your B model in the B-Toolkit using *Animator* (i.e. *anm*), how easy is it to comprehend what is happening?

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

What kind of things requires the most mental effort?

If you verify your B model in the B-Toolkit using *AutoProver*, *InterProver* or *BToolProver*, how easy is it to comprehend what is happening?

AutoProver:

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

InterProver:

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

BToolProver:

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

What kind of things requires the most mental effort?

(6) How easy is it to recognise and interpret the roles of available features in the B-Toolkit?

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

Are there any features that are particularly difficult to interpret (i.e. you are not sure what they mean and their purposes)? Which ones?

(7) Do some kinds of mistake seem particularly common or easy to make (i.e. do you often find yourself making small slips that irritate you or make you feel foolish)?

Yes **No** **Not Sure**

If **Yes**, which ones?

(8) Can you analyse, animate and verify your partially completed B model in the B-Toolkit?

Yes **No** **Not Sure**

If **No**, why?

(9) How well does the B-Toolkit allow you to play around with your B model (e.g. when you are testing your ideas/solutions, without being sure what the effects will be)?

Never				**Always**
-2	**-1**	**0**	**1**	**2**

What features of the tool help or prevent you to do this?

(10) Can you go about any task in any order you like in the B-Toolkit?

Yes **No** **Not Sure**

If **No**, why?

Does the B-Toolkit enforce you to think ahead and make certain decisions first before you could use its features?

Yes **No** **Not Sure**

If **Yes**, what decisions do you need to make in advance?

(11) Do you find any features in the B-Toolkit seem to be a strange way of working with a B model (or the B method)?

Yes **No** **Not Sure**

If **Yes**, what are they?

(12) Do you find any feature dependencies in the B-Toolkit (i.e. one feature explicitly relies upon or is determined by or uses or requires another feature)?

Yes **No** **Not Sure**

If **Yes**, how visible are the feature dependencies?

(a) Visible in both features
(b) Visible in one feature
(c) Not visible in both features

If **(b)** or **(c)**, what are the features involved? Please state them specifically.

(13) Does the B-Toolkit allow you to make notes or convey extra information beyond the B model to yourself (e.g. use comments, different fonts, colour)?

Yes **No** **Not Sure**

If **Yes**, please state the possible actions.

(14) Does the B-Toolkit insist you start the modeling task by defining or grouping things before you can do anything else?

Yes **No** **Not Sure**

If **Yes**, what sorts of things?

(15) Are there any features that seem to be similar in functionality (i.e. mean or operate similar things) but the B-Toolkit makes them appear different?

Yes **No** **Not Sure**

If **Yes**, what are they?

(16) How easy is it to become familiar with the B-Toolkit features and be able to use it in your task efficiently without referring to manuals/documentation?

Very Difficult				**Very Easy**
-2	**-1**	**0**	**1**	**2**

Why?

(17) Has the B-Toolkit helped you to find problems in your B model (i.e. are the error messages during analysis/verification easy to understand and helpful, is the animation useful etc.)?

Yes, a lot **Yes** **Yes, a little** **Not Sure** **No**

Why?

(18) How useful do you find the online documentation in the B-Toolkit (e.g. **Help** menu)?

Very Useless				**Very Useful**
-2	**-1**	**0**	**1**	**2**

Why?

(19) Has the B-Toolkit helped you to understand and learn the B method?

Yes, a lot **Yes** **Yes, a little** **Not Sure** **No**

Why?

(20) Can you think of obvious ways that the design of the B-Toolkit could be improved? What are they?

Bibliography

Abdi, H. (2007) Bonferroni and Sidak Corrections for Multiple Comparisons. In Salkind, N. J. (Eds.), *Encyclopedia of Measurement and Statistics,* Thousand Oaks, CA, Sage, 103-107.

Abrial, J. R. (1996) *The B-Method - Assigning Programs to Meanings*. Cambridge University Press, ISBN: 0521496195.

Abrial, J. R. and Mussat, L. (1998) Recent Advances in the Development and Use of the B Method. *Proceedings of the Introducing Dynamic Constraints in B, Lecture Notes in Computer Science 1393,* Springer, 83-128.

Abrial, J. R. and Cansell, D. (2003) *Click 'n' Prove – Interactive Proofs within Set Theory B*. Available online at http://www.b4free.com/ and http://www.loria.fr/cansell/cnp.html (last accessed Jan 2008).

Abrial, J. R., Butler, M., Hallerstede, S. and Voisin, L. (2006) An Open Extensible Tool Environment for Event-B. *Proceedings of the ICFEM 2006*, *Lecture Notes in Computer Science 4260,* Springer, 588-605.

Abrial, J. R. and Hallerstede, S. (2007) Refinement, Decomposition and Instantiation of Discrete Models: Application to Event-B. *Fundamentae Informatica*, 77, 1-2.

ACM (2007) *ACM Digital Library*. Available online at http://portal.acm.org (last accessed March 2008).

Agarwal R., Sinha A. and Tanniru M. (1996) Cognitive Fit in Requirements Engineering: A Study of Object and Process Models. *Journal Management Information System*, 13(2), 137–162.

Agarwal R., De P. and Sinha A. P. (1999) Comprehending Object and Process Models: An Empirical Study. *IEEE Transactions on Software Engineering,* 25(4), 541–556.

Agile (2007) *Agile Aliance*. Available online at http://www.agilealliance.com/ (last accessed Jan 2008).

Akerlind, S. G. (2002) Principles and Practices in Phenomenographic Research. *Proceedings of the International Symposium on Current Issues in Phenomenography*, 1-17.

Alexander, P. (1996) Best of Both Worlds (Formal and Semi-formal Software Engineering). *IEEE Potentials*, 14(5), 29-32.

Anderson, L. and Krathwohl, D. (2000) *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives,* Boston, MA, Longman, ISBN: 0321084057.

Babchuk, W. A. (1997) Glaser or Strauss?: Grounded Theory and Adult Education. *Midwest Research-To-Practice Conference in Adult, Continuing and Community Education*, 1-8.

Back, R. J. R. (1990) Refinement Calculus, Part II: Parallel and Reactive Programs. In de Bakker, J. W., de Roever, W. P. and Rozenberg, G. (Eds.), *Stepwise Refinement of Distributed Systems, Lecture Notes in Computer Science 430*, Springer-Verlag.

Baddeley, A. D. (1986) *Working memory*, Oxford University Press, ISBN: 0-19-852133-2.

Baddeley, A. D. (1992) Working memory. *Science,* 255, 556-559.

Baddeley, A. D. (1999) *Essentials of Human Memory.* Taylor and Francis. ISBN: 0863775454.

Basili, V. R., Selby, R. W. and Huthchens, D. H. (1986) Experimentation in Software Engineering. *IEEE Transactions on Software Engineering*, 12(7), 733-743.

Basili, V. R. and Rombach, H. D. (1988) The TAME Project: Towards Improvement Oriented Software environments. *IEEE Transactions on Software Engineering*, 14(6), 758-773.

Basili, V. R. (1992) The Experimental Paradigm in Software Engineering, *Proceedings of the International Workshop on Experimental Software Engineering*, *Lecture Notes in Computer Science 706*, Springer-Verlag.

Basili, V. R. (1996) The Role of Experimentation in Software Engineering: Past, Current, and Future. *Proceedings of the 18th International Conference on Software Engineering*, 442-449.

Basili, V. R., Shull, F. and Lanubile, F. (1999) Building Knowledge through Families of Experiments. *IEEE Transactions on Software Engineering*, 25(4), 456-473.

Batini, C., Ceri, S. and Navathe, S. (1991) *Conceptual Database Design: An Entity Relationship Approach*. Benjamin Cummings Publishing Company, ISBN: 0805302441.

Batra, D., Hoffer, J. and Bostrom, R. (1990) Comparing representations with relational and EER models. *Communications of theACM*, 33(2), 126–139.

Bauer, M. and Johnson-Laird. P. (1993) How Diagrams Can Improve Reasoning. *Psychological Science*, 4, 372-378.

Bednarik, R., and Tukiainen, M. (2004) Visual attention tracking during program debugging. *Proceedings of the 3rd Nordic Conference on Human-Computer Interaction*, 331–334.

Behm, P., Benoit, P., Faivre, A. and Meynadier, J. (1999) Météor: A Successful Application of B in a Large Project. *World Congress on Formal Methods*, *Lecture Notes in Computer Science 1709*, Springer, 369-387.

Biederman, I. (1987) Recognition-by-components: A Theory of Human Image Understanding. *Psychological Review*, 94, 115-147.

Bevan, N. (2001) International Standards for HCI and Usability. *International Journal of Human-Computer Studies*, 55(4), 533-552.

Blackwell, A. F. and Green, T. R. G. (1999) Investment of Attention as an Analytic Approach to Cognitive Dimensions. In Green, T., Abdullah, R. and Brna, P., (Eds.),

Collected Papers of the 11th Annual Workshop of the Psychology of Programming Interest Group, 24-35.

Blackwell, A. F. and Green, T. R. G. (2000) A Cognitive Dimensions Questionnaire Optimised for Users. In Blackwell, A. F. and Bilotta, E., (Eds.), *Proceedings of the 12th Annual Meeting of the Psychology of Programming Interest Group*, 137-152.

Blackwell, A. F., Whitley, K. N., Good, J. and Petre, M. (2001) Cognitive Factors in Programming with Diagrams. *Artificial Intelligence Review*, 15(1), 95-113.

Blackwell, A. and Green, T. (2003) Notational Systems – The Cognitive Dimensions of Notations Framework. In Carroll, J. M., (Eds.), *HCI Models, Theories and Frameworks: Toward a Multidisciplinary Science*, Morgan Kaufmann, ISBN: 1558608087.

Bloom, B. S. and Krathwohl, D. R. (1956) Taxonomy of Educational Objectives: The Classification of Educational Goals, by a Committee of College and University Examiners. *Handbook I: Cognitive Domain,* New York, Longmans.

Bodart, F., Sim, M., Patel, A. and Weber, R. (2001) Should Optional Properties be used in Conceptual Modelling? A Theory and Three Empirical tests. *Information System Research*. 12(4), 384-405.

Boehm, B. W., Brown, J. R. and Kaspar, J. R. (1978) Characteristics of Software Quality. *TRW Series of Software Technology*.

Boehm, B. W. and Basili, V. R. (2001) Software Defect Reduction Top 10 List. *IEEE Computer*, 135-137.

Boman, M., Bubenko, J., Johannesson, J. and Wangler, B. (1997) *Conceptual Modeling*. Prentice Hall, ISBN: 0-13-514879-0.

Bonissone, P. (1982) A Fuzzy Sets Based Linguistic Approach: Theory and Application. In Gupta, M. and Sanchez, E., (Eds.), *Approximate Reasoning in Decision Analysis*, North-Holand Publishing Company, 329-339.

Booch, G. (1994) *Object-oriented Analysis and Design with Applications*. 2nd Edition, Benjamin/Cummings, Redwood City, ISBN: 0805353402.

Booch, G., Rumbaugh, J. and Jacobson, I. (1999) *The Unified Modeling Language User Guide*. Addison-Wesley, ISBN: 0201571684.

Booch, G. (2002) Growing the UML. *Software and System Modeling,* 1(2), 157-160.

Bowen, J. P. and Hinchey, M. G. (1995) Ten Commandments of Formal Methods. *IEEE Computer*, 28(4), 56-63.

Bowen, J. (1996) *Specification and Documentation using Z: A Case Study Approach*. International Thomson Computer Press, ISBN: 1850322309.

Bowen, J. P. and Hinchey, M. G. (1997) The Use of Industrial Strength Formal Methods. *Proceedings of the 21st Annual International Conference of Computer Software and Applications*, 332-337.

Bowen, J. P. and Hinchey, M. G. (2006) Ten Commandments of Formal Methods... Ten Years Later. *IEEE Computer*, 39(1), 40-48.

Börger, E., Gargantini, A. and Riccobene, E. (2003) *Abstract State Machines 2003, Advances in Theory and Practice, 10th International Workshop ASM 2003*, *Lecture Notes in Computer Science 2589*, Springer.

Brewer, J. and Hunter, A. (1989) *Multimethod Research: A Synthesis of Styles*, Beverly Hills, CA, Sage, ISBN: 0803930771

Briand, L. C., Labiche, Y., Di Penta, M. and Yan-Bondoc, H. D. (2005) An Experimental Investigation of Formality in UML-Based Development. *IEEE Transactions on Software Engineering*, 31(10), 833-849.

Briand, L. C. (2007) Empirical Evaluation in Software Engineering: Role, Strategy, and Limitations. In Basili, V. et al., (Eds.), *Empirical Software Engineering Issues, Lecture Notes in Computer Science 4336*, Springer-Verlag Berlin Heidelberg, 21.

Britton, C. and Jones, S. (1999) The Untrained Eye: How Languages for Software Specification Support Understanding in Untrained Users. *Human Computer Interaction*, 14(1), 191-244.

Brooks, R. E. (1980) Studying Programmer Behaviour Experimentally: The Problems of Proper Methodology. *Communications of the ACM*, 23(4), 207-213.

Brooks, R. (1983) Towards a Theory of the Comprehension of Computer Programs. *International Journal of Man- Machine Studies*, 18, 543-554.

Brun, P. and Beaudouin-Lafon, M. (1995) A Taxonomy and Evaluation of Formalisms for the Specification of Interactive Systems. *People and Computers X*, Cambridge University Press, 197-212.

Burton-Jones A. and Meso, P. (2002) How Good are these UML Diagrams? An Empirical Test of the Wand and Weber Good Decomposition Model. *Proceedings of the 23rd International Conference on Information Systems*, 101-144.

Butler, M., Leuschel, M. and Snook, C. (2005) Tools for System Validation with B Abstract Machines. *Proceedings of the 12th International Workshop on Abstract State Machines*, 57-69.

B-Core (1999) *B-Toolkit*, *On-line Manual*, B-Core Limited, Oxon, UK. Available online at http://www.b-core.com/ONLINEDOC/Contents.html (last accessed Jan 2008).

Cansell, D., Hallerstede, S. and Oliver, I. (2004) UML-B Specification and Hardware Implementation of a Hamming Coder/Decoder. In Mermet, J., (Eds.), *UML-B Specification for Proven Embedded Systems Design*, Springer, Chapter 13.

Carew, D., Exton, C. and Buckley, J. (2005) An Empirical Investigation of the Comprehensibility of Requirements Specifications. *International Symposium on Empirical Software Engineering*, 256-265.

Carver, J., Jaccheri, L., Morasca, S. and Shull, F. (2003) Issues in Using Students in Empirical Studies in Software Engineering Education. *Proceedings of the 9th International Software Metrics Symposium*, 239-249.

Cassell, C. and Symon, G. (Eds.), (1994) *Qualitative Methods in Organizational Research.* Thousand Oaks, CA, Sage, ISBN: 0803987706.

Casset, L. (2002) Development of an Embedded Verifier for Java Card Byte Code using Formal Methods. *Formal Methods Europe (FME'02)*, *Lecture Notes in Computer Science 2391*, Springer-Verlag.

Chambliss, M. J and Calfee, R. C. (1998) *Textbooks of Learning: Nurturing Children's Minds*, Wiley-Blackwell, ISBN: 1557864128.

Chandler, P. and Sweller, J. (1991) Cognitive Load Theory and the Format of Instruction. *Cognition and Instruction,* 8, 293-332.

Chandler, P. and Sweller, J. (1992) The Split-attention Effect as a Factor in the Design of Instruction. *British Journal of Educational Psychology*, 62, 233-246.

Chen, P. (1976) The Entity-Relationship Model: Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1), 9-37.

Clark, J. M. and Paivio, A. (1991) Dual Coding Theory and Education. *Educational Psychology Review,* 53(2), 445-459.

Clarke, E. M., Grumberg, O. and Peled. D. (1999) *Model Checking*, MIT Press, ISBN: 0-262-03270-8.

Clarke, S. (2001) Evaluating a New Programming Language. *Proceedings of the 13th Annual Meeting of the Psychology of Programming Interest Group*, 275-289.

ClearSy (2003) *AtelierB User Manual V3.6*, ClearSy System Engineering, Aix-en-Provence, France.

Cohen, J. (1988). *Statistical Power Analysis for the Behavioural Sciences*. 2nd edition, Hillsdale, NJ, Lawrence Earlbaum Associates, ISBN 0-8058-0283-5.

Cook, T. D. and Campbell, D. T. (1979a) *Experimental and Quasi-experimental Designs for Research*. Boston MA, Boughton Mifflin Company.

Cook, T. D. and Campbell, D. T. (1979b) *Quasi-Experimentation: Design and Analysis for Field Settings*. Chicago, Illinois, Rand McNally.

Cook, L. K. and Mayer, R. E. (1988) Teaching Readers about the Structure of Scientific Text. *Journal of Educational Psychology*, 80, 448-456.

Cook, S., Kleepe, A., Mitchell, R., Rumpe, B., Warmer, J. and Willis, A. (2001) The Amsterdam Manifesto on OCL. In Clark T. and Warmer J. (Eds.), *Advances in Object Modelling with the OCL*, *Lecture Notes in Computer Science 2263*, Springer, 115-149.

Corritore, C. and Wiedenbeck, S. (2001) An Exploratory Study of Program Comprehension Strategies of Procedural and Object-oriented Programmers. *International Journal of Human-Computer Studies*, 54(1), 1-23.

Courtney, R. E and Gustafson, D. A. (1992) Shotgun Correlations in Software Measures. *Software Engineering Journal*, 8(1), 5-13.

Cox, K. (2000) Cognitive Dimensions of Use Cases: Feedback from a Student Questionnaire. In Blackwell, A. F. and Bilotta, E. (Eds.), *Proceedings of the 12th Annual Meeting of the Psychology of Programming Interest Group*, 99-122.

Craigen, D., Gerhart, S. and Ralston, T. (1995) *Applications of Formal Methods*. In Hinchey, M., Bowen, J., (Eds.), Englewoodcliffs, NJ, Prentice-Hall, ISBN:0133669491.

Creswell, J. W. (2002) *Research design: Qualitative, Quantitative and Mixed Methods Approache*s, 2nd Edition, Sage Publications, ISBN: 0761924426.

Crosby, M. and Stelovsky, J. (1990) How Do We Read Algorithms? A Case Study. *IEEE Computer*. 23(1), 24-35.

Cunniff, N. and Taylor R. P. (1987) Graphical vs Textual Representation: An Empirical Study of Novices' Program Comprehension. *Empirical Studies of Programmers: 2nd Workshop*, 114-131.

Curtis, B., Sheppard, S. B., Kruesi-Bailey, E., Bailey, J. and Boehm-Davis, D. A. (1989) Experimental Evaluation of Software Documentation Formats. *Journal of System and Software*, 9(2), 167-207.

Daly, J. (1996) *Replication and a Multi-Method Approach to Empirical Software Engineering Research*. PhD thesis, University of Strathclyde, Glasgow, UK.

Dascalu, S. and Hitchcock, P. (2002) Software Engineering: Theory, Application and Practice: An Approach to Integrating Semi-Formal and Formal Notations in Software Specification. *Proceedings of the ACM Symposium on Applied Computing*, 1014-1020.

Davis, A., Overmyer, S., Jordan, K., Caruso, J., Dandashi, F., Dinh, A., Kincaid, G., Ledeboer, G., Reynolds, P., Sitaram, A., Ta, A. and Theofanos, M. (1993) Identifying and Measuring Quality in a Software Requirements Specification. *Proceedings of the 1st. International Software Metrics Symposium*, *IEEE*, 141-152.

DeGrace, P. and Stahl, L. H. (1998) *Wicked Problems, Righteous Solutions: A Catalogue of Modern Engineering Paradigms,* Prentice Hall.

Denzin, N. and Lincoln, Y. (1994). *Handbook of Qualitative Research.* Thousand Oaks, CA, Sage.

Easterbrook, S. M., Singer, J., Storey, M. and Damian, D. (2008) Selecting Empirical Methods for Software Engineering Research. In Shull, F., Singer, J. and Sjøberg, D. I. K., (Eds.), *Guide to Advanced Empirical Software Engineering*, 12(388), 37.

Ebert C. (1997) The Road to Maturity: Navigating between Craft and Science. *IEEE Software*, 77-82.

Eclipse (2007) *Eclipse – An Open Development Platform*. Available online at http://www.eclipse.org/ (last accessed Jan 2008).

Electronics Computer Science (ECS) (2007) *COMP3011 Critical Systems*. Available online at http://www.ecs.soton.ac.uk/syllabus/COMP3011.html (last accessed Jan 2008).

Efron, B. (1981) Nonparametric Standard Errors and Confidence Intervals. *Canadian Journal of Statistics*, 36, 369-401.

Efron, B. and Tibshirani, R. (1986) The Bootstrap Method for Standard Errors, Confidence Intervals and Other Measures of Statistical Accuracy. *Statistical Science*, 1(1), 1-35.

Efron, B. and Tibshirani, R. (1993) *An Introduction to the Bootstrap*. Chapman and Hall.

Elmasri, R and Navathe, S. (2001) *Fundamentals of Database Systems*. 3rd Edition, Addison-Wesley, ISBN: 0805317554.

Farbey, B. (1993) Software Quality Metrics: Considerations about Requirements and Requirement Specifications. In Thayer, R. and McGetteric, A., (Eds.), *Software Engineering: A European Perspective*, Los Alamitos, CA, IEEE Computer Society Press, 138-142.

Fenton, N., Pfleeger, S. L. and Glass, R. L. (1994) Science and Substance: A Challenge to Software Engineers. *IEEE Software*, 11(4), 86-95.

Fenton, N. E. and Pfleeger, S. L. (1996) *Software Metrics: A Rigorous and Practical Approach*. Thomson Computer Press, ISBN: 0534954251.

Finney, K. and Fedorec, A. (1996a) An Empirical Study of Specification Readability Teaching and Learning Formal Methods, In Dean, N. and Hinchey, M., (Eds.), *Teaching and Learning Formal Methods*, New York, Academic Press, ISBN: 0123490405.

Finney, K. (1996b) Mathematical Notation in Formal Specification: Too difficult for the Masses? *IEEE Transactions on Software Engineering*, 22(2), 158-159.

Finney, K., Fenton, N. and Fedorec, A. (1999) Effects of Structure on the Comprehensibility of Formal Specifications, *IEE Proceedings- Software Engineering*, 146(4), 193-202.

Firesmith, D., Henderson-Sellers, B. and Graham, I. (1998) *The Open Modelling Language (OML) Reference Manual*. SIGS Book, Cambridge University Press.

Fisher, R. A. (1956) Statistical Methods and Scientific Inference. In Bennet, J. H., (Eds.), (1990), *Statistical Method, Experimental Design and Statistical Inference*, Oxford University Press, ISBN: 0198522290.

Fitzgerald, J. S. and Larsen, P. G. (1998) *Modelling Systems: Practical Tools and Techniques in Software Engineering.* Cambridge University Press, ISBN 0-521-62348-0.

Formal Methods Virtual Library (2007) Available online at http://vl.fmnet.info/ (last accessed Jan 2008).

Foster, J. (1991) Program Lifetime: A Vital Statistic for Maintenance. *Proceedings of the IEEE Conference on Software Maintenance*, 98-103.

Fowler, M. (2004) *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 3rd Edition, Addison-Wesley, ISBN: 9780321193681.

Gage, D. and McCormick, J. (2004) *Why Software Quality Matters-Case 109: A Dissection*, Baseline, 34-59. Available online at http://www.eweek.com/article2/ (July 2006).

Gauch, R. R. (2000) *Statistical Methods for Researchers Made Very Simple*. Lanham Md, University Press of America, ISBN: 0761817018.

Geigy (1982) Scientific Tables. In Lentner, C., Diem, K. and Seldrup, J. (Eds.), *Volume 2: Introduction to Statistics, Statistical Tables, Mathematical Tables*. 8th Edition, CIBA-GEIGY Ltd.

Gemino, A. (1998) To Be or May to Be: An Empirical Comparison of Mandatory and Optional Properties in Conceptual Modelling. *Proceedings of the Annual Conference of the Administrative Sciences Association of Canada, Information Systems Division,* 19(4), 33-44.

Gemino, A. and Wand, Y. (2003) Evaluating Modelling Techniques based on Models of Learning. *Communications of the ACM*, 46(10), 79-84.

Gemino, A. (2004) Empirical Comparisons of Animation and Narration in Requirements Validation. *Requirements Engineering Journal,* 9, 153-168.

Gemino, A. and Wand, Y. (2005) Complexity and Clarity in Conceptual Modelling: Comparison of Mandatory and Optional Properties. *Data and Knowledge Engineering,* 55, 301-326.

Gerhart, S., Craigen, D. and Ralston, T. (1993) Observation on Industrial Practice Using Formal Methods. *Proceedings of 15th International Conference of Software Engineering*, IEEE Computer Science Press, 24-33.

Gerhart, S., Craigen, D. and Ralston, T. (1994) Experience with Formal Methods in Critical Systems. *IEEE Software*, 21-28.

Gilgun, J. F, Daly, K. and Handel, G. (Eds.), (1992) Qualitative Methods in Family Research*,* Thousand Oaks, CA, Sage.

Glaser, B. G. and Strauss, A. L. (1967) *The Discovery of Grounded Theory: Strategies for Qualitative Research*, London, Weidenfeld and Nicolson, Reprint (1999), ISBN: 0202302601.

Glaser, B (1992) *Basics of Grounded Theory Analysis: Emergence vs. Forcing.* Sociology Press. Mill Valley, California.

Glass, R. J, Vessey, I. and Ramesh, V. (2002) Research in Software Engineering: An Analysis of the Literature. *Information and Software Technology*, 44, 491-506.

Glezer, C., Last, M., Nachmany, E. and Shoval, P. (2005) Quality and Comprehension of UML Interaction Diagrams - An Experimental Comparison. *Information and Software Technology*, 47, 675-692.

Godo, L., de Mantaraz, L. R., Sierra, C. and Verdaguer, A. (1989) MILORD: The Architecture and Management of Linguistically Expressed Uncertainty. *International Journal of Intelligent Systems*, 4, 471-501.

Gogolla, M. and Richters, M. (2001) Expressing UML Class Diagrams Properties with OCL. In Clark T. and Warmer J., (Eds.), *Advances in Object Modelling with the OCL*, *Lecture Notes in Computer Science 2263*, Springer, Berlin, 85-114.

Gomaa, H. (2000) *Designing Concurrent, Distributed, and Real-Time Applications with UML*. 1st Edition*,* Addison-Wesley Professional, ISBN: 0201657937.

Gosset, W. S., Pearson, E. S. and Wishart, J. (1942) *Student's Collected Papers*. Biometrika Office, London.

Grady, R. B. and Caswell, D. L. (1987) *Software Metrics: Establishing a Company Wide Program*. Prentice-Hall.

Green, T. (1980) Programming as a Cognitive Activity. In Smith, H. and Green, T., (Eds.), *Human Interaction with Computers*, Academics, London, 271-320.

Green, T. R. G. (1989) Cognitive Dimensions of Notations. In Sutcliffe, A. and Macaulay, L. (Eds.), *People and Computers V*, Cambridge University Press, Cambridge, 443-460.

Green, T. R. G. and Petre, M. (1996) Usability Analysis of Visual Programming Environments: a Cognitive Dimensions Framework. *Journal of Visual Languages and Computing*, 7, 131-174.

Green, T. R. G. and Blackwell, A. F. (1998) Design for Usability using Cognitive Dimensions, *Tutorial session at British Computer Society conference on Human Computer Interaction*.

Gries, D. (1981) *The Science of Programming*. Springer-Verlag, ISBN: 0387964800.

Gubrium, J. F. and Sankar, A. (1994) *Qualitative methods in Aging Research.* Thousand Oaks, CA, Sage.

Gurevich,Y., Kutter, P. W., Odersky, M. and Thiele, L. (2000) *Abstract State Machines - Theory and Applications: International Workshop of ASM 2000,* 1st edition, Monte Verita, Switzerland, Springer, ISBN: 3540679596.

Hall, A. J. (1996) Using Formal Methods to Develop an ATC Information System. *IEEE Software*, 13(2), 66-76.

Hallerstede, S. (2003) Parallel Hardware Design in B. In *3rd International Conference of B and Z Users (ZB'03)*, *Lecture Notes in Computer Science 2651*, Springer-Verlag, 101-102.

Hallerstede, S. (2006) Justifications for the Event-B Modelling Notation. B 2007. *Lecture Notes in Computer Science* 4355, Springer, 49-63.

Harry, A. (1997) *Formal Methods Fact File: VDM and Z*. Wiley, ISBN: 0471958573.

Henniker, R., Hussmann, H. and Bidoit, M. (2001) On the Precise Meaning OCL Constraints, In Clark T. and Warmer J. (Eds.), *Advances in Object Modelling with the OCL*, *Lecture Notes in Computer Science 2263*, Springler, Berlin, 69-84.

Hinchey, M. G. and Bowen, J. P. (1999) *Industrial-Strength Formal Methods in Practice*. FACIT Series, Springer-Verlag, ISBN: 1852336404.

Hinchey, M. G. (2002) Confessions of a Formal Methodist. *Proceedings of the 7th Australian Workshop on Safety-Critical Systems and Software (SCS 02)*, In Lindsay, P., (Eds.), *Conferences in Research and Practice in Information Technology*, Australian Computer Society, 15, 17-20.

Hoosain, R. (1983) Memorization of Classical Chinese. *Psychologia: An International Journal of Psychology in the Orient,* 26, 193-197.

Houston, I. and King, S. (1991) CICS Project Report: Experiences and Results from the Use of Z. *Proceedings of the VDM '91: Formal Development Methods*, *Lecture Notes in Computer Science 551*, Springer-Verlag, New York.

Höfer, A., Tichy, W. F. and Basili, V. et al. (Eds.) (2007) Status of Empirical Research in Software Engineering. *Empirical Software Engineering Issues*, *Lecture Notes in Computer Science* 4336, Springer-Verlag Berlin Heidelberg, 10-19.

Höst, M, Regnell, B and Wohlin, C. (2000) Using Students as Subjects- A Comparative Study Of Students & Professionals in Lead-Time Impact Assessment, *Empirical Software Engineering*, Vol.5, No.3, pp. 201-214.

Huitt, W. (1992) Problem Solving and Decision Making: Consideration of Individual Differences using the Myers-Briggs Type Indicator. *Journal of Psychological Type,* 24, 33-44. Available online at http://chrion.valdosta.edu/whuitt//papers/prbsmbti.html (last accessed Jan 2008).

Huitt, W. (2004) Taxonomy of the Cognitive Domain. In Bloom et al. (Eds.), *Educational Psychology Interactive,* Valdosta, GA, Valdosta State University. Available online at http://chiron.valdosta.edu/whuitt/col/cogsys/bloom.html (last accessed Jan 2008).

Hummel, J. and Huitt, W. (1994) What you measure is what you get, *GaASCD Newsletter: The Reporter*. 10-11. Available online at http://teach.valdosta.edu/whuitt/papers/wymiwyg.html (last accessed Jan 2008).

Idani, A. and Ledru, Y. (2006) Dynamic Graphical UML Views from Formal B Specifications. *Information and Software Technology*, 48(3), 154-169.

IEEE (1987) *Software Engineering Standards*. The Institute of Electrical and Electronics Engineers.

IEEE (1990) *IEEE Standard Glossary of Software Engineering Terminology*. Institute of Electrical and Electronics Engineers, Inc., Standard 610.12-1990.

IEEE (1998) *IEEE Recommended Practice for Software Requirements Specifications*. Software Engineering Standards Committee of the IEEE Computer Society, IEEE Std 830-1998.

IEEE (2007) *IEEE Xplore*. Available online at http://ieeexplore.ieee.org (last accessed March 2008).

Insightful (2006) *SPLUS*. Available online at http://www.insightful.com/products/splus/default.asp (last accessed July 2007).

Irani, P. and Ware, C. (2000) Diagrams Based on Structural Object Perception. *Advanced Visual Interfaces*, 61-67.

ISO 9126-1 (2001) *Software Engineering, Product Quality - Part I: Quality Model*, International Organisation for Standardisation.

ISO 9000 and ISO 14000 (2004). Available online at http://www.iso.org/iso/en/iso9000-14000/index.html (last accessed July 2007)

ISO 9126-3 (2003) *Software Engineering, Product Quality - Part 3: Internal Metrics*, International Organisation for Standardisation.

ISO 9126-4 (2004) *Software Engineering, Product Quality - Part 4: Quality in use metrics*, International Organisation for Standardisation.

Jacobson, I., Jonsson, P. and Overgaard, G. (1992) *Object-Oriented Software Engineering: A Use Case Driven Approach*. ACM Press/Addison-Wesley, Reading.

Jarvenpaa, S. and Machesky, J. (1989) Data Analysis and Learning: An Experimental Study of Data Modeling Tools. *International Journal of Man-Machine Studies,* 31, 367–391.

Jeffery, R. and Scott, L. (2002) Has Twenty-five Years of Empirical Software Engineering Made a Difference?, *Proceedings of the 9th Asia-Pacific Software Engineering Conference*, 539-546.

Jones, C. B. (1990) *Systematic Software Development using VDM*. Prentice Hall International.

Jones, B. and Kenward, M. G. (2003) *Design and Analysis of Cross-over Trials*. 2nd Edition, Chapman and Hall, London, ISBN: 0412606402.

Juristo, N. and Moreno, A. M. (2001) *Basics of Software Engineering Experimentation*.1st Edition, Springer, ISBN-10: 0792379904.

Kamsties, E., Von Knethen, A. and Reussner, R (2003) A Controlled Experiment to Evaluate How Styles Affect the Understandability of Requirements Specifications. *Information and Software Technology*, 45, 955-965.

Keppel, G. (1991) *Design and Analysis: A Researcher's Handbook*. 3rd Edition, Prentice Hall.

Kienle, H. M. and Muller, H. A. (2007) Requirements of Software Visualization Tools: A Literature Survey. *Proceedings of the 4th IEEE International Workshop Visualizing Software for Understanding and Analysis*, 2-9.

Kim, J. Hahn, J. and Hahn, H. (2000) How do we Understand a System with (So) Many Diagrams? Cognitive Integration Processes in Diagrammatic Reasoning. *Information Systems Research,* 11(3), 284-303.

Kim, S. K. and Carrington, D. (2004) A Formal Object-Oriented Approach to Defining Consistency Constraints for UML Models. *Proceedings of Australian Software Engineering Conference*, 87-94.

Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., El Emam, K. and Rosenberg, J. (2002a) Preliminary Guidelines for Empirical Research in Software Engineering. *IEEE Transactions on Software Engineering*, 28(8), 721-734.

Kitchenham, B. and Pfleeger, S. L. (2002b) Principles of Survey Research: Parts 1-6. *ACM SIGSOFT Software Engineering Notes*, 27(1-6).

Kitchenham, B., Budgen, D., Breton, P., Turner, M., Charters, S. and Linkman, S. (2007) Large Scale Software Engineering Questions – Expert Opinion or Empirical Evidence?. *IET Software*, 1(5), 161-171.

Koch, G. G. (1972) The Use of Non-parametric Methods in the Statistical Analysis of the Two-period Change-over Design. *Biometrics*, 28, 577-584.

Koenemann, J. and Robertson, S. (1991) Expert Problem Solving Strategies for Program Comprehension. *Proceedings of HCI'91*, ACM, New York, 125-130.

Koffka, K. (1935) *Principles of Gestalt Psychology*. London, Kegan Paul, Trench.

Krogstie, J. (1998) Integrating the Understanding of Quality in Requirements Specification and Conceptual Modeling. *ACM SIGSOFT Software Engineering Notes*, 23(1), 86-91.

Krupp, A., Lundkvist, O., Schattkowsky, T. and Snook, C. (2004) Adaptive Cruise Controller Case Study. In Mermet, J. (Eds.), *UML-B Specification for Proven Embedded Systems Design*, Springer, Chapter 12.

Kruskal, W. H and Wallis, W. A. (1952) Use of Ranks in One-criterion Variance Analysis. *Journal of the American Statistical Association*, 47(260), 583–621.

Kung, C. H. and Solvberg, A. (1986) Activity Modelling and Behaviour Modelling. In Olle, T.W., Sol, H.G. and Verrijn-Stuart, A.A. (Eds.), *Information Systems Design Methodologies: Improving the Practice*, IFIP, Amsterdam, North-Holland, 145-171.

Kutar, M., Britton, C. and Barker, T. (2002) A Comparison of Empirical Study and Cognitive Dimensions Analysis in the Evaluation of UML Diagrams. In Kuljis, J., Baldwin, L. and Scoble, R. (Eds.), *Proceedings of the 14th Annual Meeting of the Psychology of Programming Interest Group*, 1-14

Lano, K., Clark, D. and Androutsopoulos, K. (2004) UML to B: Formal Verification of Object-Oriented Models. In Boiten, E. A., Derrick, J. and Smith, G., (Eds.), *Proceedings of the 4th International Conference on Integrated Formal Methods*, *Lecture Notes in Computer Science 2999,* Springer-Verlag, 187-206.

Lau, K. and Banach, R. (2005) *Formal Methods and Software Engineering. Lecture Notes in Computer Science 3785*, Springer-Verlag.

Larkin, J. H. and Simon, H.A. (1987) Why a Diagram Is (Sometimes) Worth Ten Thousand Words. *Cognitive Science*, 11, 65-99.

Larman, C. (2004) *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. 3rd Edition, Prentice Hall PTR, ISBN-10: 0131489062.

Ledang, H. and Souquieres, J. (2002) Contributions for Modeling UML Statecharts in B, *Proceedings of the Third International Conference on Integrated Formal Methods (IFM'2002)*, *Lecture Notes in Computer Science 2335*, Springer-Verlag, 109-127.

Leuschel, M. and Butler, M. (2003) ProB: A Model Checker for B. In Keijiro, A., Gnesi, S. and Dino, M., (Eds.), *Proceedings of the Formal Methods Europe 2003 2805*, 855-874.

Leuschel, M. and Butler, M. (2005) Automatic Refinement Checking for B. *Proceedings of ICFEM'05, Lecture Notes in Computer Science 3785*, Springer-Verlag.

Leuschel, M. and Butler, M. (2006) *ProB: An Automated Analysis Toolset for the B Method*, Technical Report, Electronics and Computer Science, University of Southampton.

Levin, H. (1983) *Cost-effectiveness: A Primer.* Thousand Oaks, CA, Sage Publications.

Lim, K. H. and Benbasat, I. (2002) The Influence of Multimedia on Improving the Comprehension of Organizational Information. *Journal of MIS*, 19(1), pp. 99–127.

Lindley, D. V. and Scott, W. F. (1984) *New Cambridge Elementary Statistical Table.* Cambridge University Press, Cambridge.

Littman, D. C., Pinto, J., Letovsky, S. and Soloway, E. (1986) Mental Models and Software Maintenance. In Soloway, E. and Iyengar, S., (Eds.), *Empirical Studies of Programmers*, Ablex, Norwood. N.J, 80-98.

Loucopoulus, P. and Karakostas, V (1995) *Systems Requirements Engineering*. McGraw-Hill.

Lukowicz, P., Heinz, E. A., Prechelt, L. and Tichy, W. F. (1995) Experimental Evaluation in Computer Science: A Quantitative Study. *Journal of Systems and Software*, 28(1), pp. 9-18.

Macaulay, L. (1996) Requirements for Requirements Engineering Techniques. *Proceedings of the Second International Conference on Requirements Engineering*, IEEE Computer Society Press, Los Alamitos CA, 157-164.

MacKenzie, D. (2001) *Mechanizing Proof: Computing, Risk, and Trust,* MIT Press.

MacKinlay, J. D. (1986) Automating the Design of Graphical Presentation of Relational Information. *ACM Transaction on Graphics*, 5(2), 110-141.

Martin, J. (1982) A Garbage Can Model of the Research Process. In McGrath, J., Martin, J. and Kulka, R. (Eds.), *Judgment Calls in Research*, Sage, Beverly Hills, CA.

Martin, S. (2003) The Best of Both Worlds Integrating UML with Z for Software Specifications. *Journal of Computing and Control Engineering*, 14(2), 8-11.

MATISSE (2002) *Methodologies and Technologies for Industrial Strength Systems Engineering*. Available online at http://www.matisse.qinetiq.com/ (last accessed July 2006)

Mayer, R.E. (1989a) Models for Understanding. *Review of Educational Research,* 59(1), 43–64.

Mayer, R. E. (1989b) Systematic Thinking Fostered by Illustrations in Scientific Text, *Journal of Educational Psychology*, 81, 240-246.

Mayer, R. E. and Gallini, J. K. (1990) When is an Illustration Worth Ten Thousand Words?. *Journal of Educational Psychology*, 82, 715-726.

Mayer, R. E. and Anderson, R. B. (1991) Animations Need Narrations: An Experimental Test of a Dual-Coding Hypothesis. *Journal of Educational Psychology*, 83, 484-490.

Mayer, R. E. and Anderson, R. B. (1992) The Instructive Animation: Helping Students Build Connections between Words and Pictures in Multimedia Learning. *Journal of Educational Psychology*, 84, 444-452.

Mayer, R. E., Bove, W., Bryman, A., Mars, R. and Tapangco, L. (1996) When Less is More: Meaningful Learning from Visual and Verbal Summaries of Science Textbook Lessons. *Journal of Educational Psychology*, 88, 64-73.

Mayer, R.E. (1999) *The Promise of Educational Psychology*, Upper Saddle River, Prentice Hall.

Mayer, R.E. (2001) *Multimedia Learning*, Cambridge University Press.

McCall, J., Richards, P. and Walters, G. (1977) *Factors in Software Quality*. 1, 2, 3.

McCluskey, T., Porteous, J., Naik, Y., Taylor, C., and Jones, S. (1995) A Requirements Capture Method and its use in an Air Traffic Control Application. *Software Practice and Experience*, 25, 47-73.

McDermid, J. (1993) Safety-Critical Software: A Vignette. *IEE Software Engineering Journal*, 1, 2-3.

McGrath, J. E. (1995) Methodology Matters: Doing Research in the Behavioral and Social Sciences. In Baecker, R. M., Grudin, J., Buxton, W. A. and Greenberg, S. (Eds.), *Human-Computer interaction: Toward the Year 200*0, Morgan Kaufmann Publishers, San Francisco, CA, 152-169.

Meyer, E. and Souquieres, J. (1999) A Systematic Approach to Transform OMT Diagrams to a B Specification. *Proceedings of the Worlds Congress on Formal Methods in the Development of Computing Systems (FM'99*), *Lecture Notes in Computer Science 1708*, Springer-Verlag, 875-895.

Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. (1986), *Machine Learning: An Artificial Intelligence Approach, Volume II*.Morgan Kaufmann, ISBN 0-934613-00-1.

Microsoft (2007) *Microsoft Corporation.* Available online at http://www.microsoft.com (last accessed Sept 2007).

Miles, M. B. and Huberman, A. M. (1994) *Qualitative Data Analysis*. 2nd Edition, London, Sage Publications.

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, R.G. and Jr. (1981) *Simultaneous Statistical Inference*. 2nd Edition, New York, Springer-Verlag.

Mishan, E. J. (1988) *Cost-Benefit Analysis*. 4th Edition, London, Unwin Hyman.

Moody, L., Shanks, G. and Darke, P. (1998) Improving the Quality of Entity Relationship Models–Experience in Research and Practice. *Proceedings of the 17th International Conference on Conceptual Modelling*, 255-276.

Moore, D. S and McCabe, G. P. (2006) *Introduction to the Practice of Statistics*. 5th Edition, W. H. Freeman, New York, ISBN: 071676282.

Moore, G. (1991) *Crossing the Chas*m. Harper Business, New York.

Morse, J. M. and Field, P. A. (1995). *Qualitative Research Methods for Health Professionals*. 2nd Edition, Thousand Oaks, CA, Sage, ISBN: 0803973276.

Motschnig-Pitrik, R. (1993) The Semantics of Parts versus Aggregates in Data/Knowledge Modelling. *Proceedings of the CaiSE'93*, *Lecture Notes in Computer Science 685*, Springer-Verlag, 352-373.

Mylopoulos, J. (1992) Conceptual Modelling and Telos. In Loucopoulos, P. and Zicari, R. (Eds.), *Conceptual Modeling,Databases, and CASE: An Integrated View of Information Systems Development*, John Wiley and Sons Inc., New York.

NASA (1998) *Formal Methods Specification and Verification Guidebook*. Planning and Technology Insertion, [NASA-GB-001-97], 1, Release 1.0, National Aeronautics and Space Administration, Washington DC.

Nielsen, J. (1994) *Usability Engineering*. Morgan Kaufmann Publishers, ISBN: 0125184069.

Nosek, J. and Ahrens, J. (1986) An Experiment to Test User Validation of Requirements: Data Flow Diagrams vs. Task Oriented Menus. *International Journal of Man-Machine Studies*, 25, 675–684.

Object Management Group (OMG) (2006) Introduction to OMG's Unified Modeling Language (UML). Available online at http://www.omg.org/gettingstarted/what_is_uml.htm (last accessed Jan 2008).

Olive, A. (2000) An Introduction to Conceptual Modeling of Information Systems. In Piattini, M. and Diaz, O. (Eds.), *Chapter 2: Advanced Database Technology and Design*, Artech House, 25-57.

Onwuegbuzie, A. J. and Teddlie, C. (2003) A Framework for Analyzing Data in Mixed Methods Research. In Tashakkori A. and Teddlie, C. (Eds.), *Handbook of mixed methods in social and behavioral research*, Thousand Oaks, CA, Sage, 351-383.

Onwuegbuzie, A. J. and Leech, N. L. (2005) Taking the "Q" out of research: Teaching Research Methodology Courses Without the Divide between Quantitative and Qualitative Paradigms. *Quality & Quantity: International Journal of Methodology,* 39(3), 267-295.

OTI (2003) *Eclipse Platform Technical Overview*, Object Technology International Inc. Available online at http://www.eclipse.org/ (last accessed July 2006)

Page-Jones, M. (1980) *The Practical Guide to Structured Systems Design*. Yourdon Press, ISBN: 0917072170.

Paivio, A. (1986) *Mental Representation: A Dual Coding Approach*. Oxford University Press, ISBN: 0195066669.

Palmer, S. E. (1992) Common Region: A New Principle of Perceptual Grouping, *Cognitive Psychology,* 24, 436–447.

Palmer, S. E. and Rock, I. (1994) Rethinking Perceptual Organization: The Role of Uniform Connectedness. *Psychonomic Bulletin and Review,* 1, 29–55.

Parnas, D. L. (2003) *Design through Documentation: The Path to Software Quality*. Available online at http://www.csis.ul.ie/Research/ParnasLectures/ (last accessed July 2007)

Patton, M. Q. (1990). *Qualitative Research and Evaluation Methods*. 2nd Edition. Newbury Park, CA, Sage.

Pender, T. (2003) *UML Bible*. Wiley, ISBN: 0764526049.

Penney, C.G. (1989) Modality Effects and the Structure of Short Term Verbal Memory. *Memory and Cognition,* 17, 398-422.

Pennington, N. (1987) Comprehension Strategies in Programming. In Olson, G., Sheppard, S. and Soloway, E. (Eds.), *Empirical Studies of Programmers: 2nd Workshop*, Ablex, Norwood NJ, 100-113.

Perry, D. E., Porter, A. A. and Votta, L. G. (2000) Empirical Studies of Software Engineering: A Roadmap. *Proceedings of the Conference on the Future of Software Engineering*, ACM Press, 345-355.

Petre, M. (1995) Why Looking isn't Always Seeing: Readership Skills and Graphical Programming. *Communications of the ACM*, 38(6), 33-44.

Petre, L., Troubitsyna, E., Waldén, M., Boström, P., Engblom, N. and Jansson, M. (2001) *Methodology of Integration of Formal Methods within a Healthcare Case study*, TUCS Technical Reports, No.436, Turku Centre for Computer Science, Finland.

Pfleeger, S. L. (1995) Experimental Design and Analysis in Software Engineering: Part 1-5. *ACM SIGSOFT Software Engineering Notes*, 20(1-5).

Pfleeger, S. L. and Hatton, L. (1997) Investigating the influence of formal methods. *IEEE Computer*, 30(2), 33-43.

Pfleeger, S. L. (1999) Albert Einstein and Empirical Software Engineering. *IEEE Computer*, 32(10), 32-38.

Pfleeger, S. L. and Menezes, W. (2000) Marketing Technology to Software Practitioners. *IEEE Software*, 17(1), 27-33.

Piattini, M., Genero, M., Poels, G. and Nelson, J. (2005) Towards a Framework for Conceptual Modelling Quality. In Genero, M., Piattini, M. and Calero, C. (Eds.), *Metrics for Software Conceptual Models*, London : Imperial College Press, ISBN: 1860944973.

Pigoski, T.M. (1996) Practical Software Maintenance: Best Practices for Managing your Software Investment. Wiley Computer Publishing, ISBN: 0471170011.

Plat, N., van Katwijk, J. and Toetenel, H. (1992) Application and Benefits of Formal Methods in Software Development. *Software Engineering Journal*, 7(5), 335-346.

Pollock, E. J. (2000) *Assimilating Complex Information*. PhD Thesis, University of New South Wales, Australia.

Pouzancre, G. (2003) How to Diagnose a Modern Car with a Formal B model? *Formal Specification and Development in Z and B*, *Lecture Notes in Computer Science 2651*, Springer-Verlag, 98-100.

PUSSEE (2003) *Paradigm Unifying System Specification Environments for Proven Electronic Design*. Available online at http://www.keesda.com/pussee/ (last accessed July 2006)

Rainer, A., Jagielska, D. and Hall, T. (2005) Software Practice versus Evidence-based Software Engineering Research. *Proceedings of the Workshop on Realising Evidence-based Software Engineering*, ICSE, ACM Press, 1-4.

Ramsey, R., Atwood, M. and van Doren J. (1993) Flowcharts versus Program Design Languages: An Experimental Comparison. *Communications of the ACM*, 26(6), 445–449.

Rational (2000) *Rose extensibility reference – Rational Rose 2000e*. Rational Software Corporation, Part Number 800-023329-000.

Rayner, K. (1998) Eye movements in Reading and Information Processing: 20 Years of Research. *Psychological Bulletin,* 124, 372–422.

RODIN (2004) *Rigorous Open Development Environment for Complex Systems*. Available online at http://rodin.cs.ncl.ac.uk/ (last accessed July 2006)

Romero, P., Lutz, R., C0x, R. and Du Boulay, B. (2002) Co-ordination of Multiple External Representations during Java Program Debugging. *Proceedings of the IEEE 2002 Symposia on Human Centric Computing Languages and Environments,* IEEE Computer Society, 207.

Rosenberger, W. F. (1996) Dealing with Multiplicities in Pharmacoepidemiologic Studies. *Pharmacoepidemiology and Drug Safety*, 5, 95-100.

Rosenthal, R. (1991) *Meta-analytic Procedures for Social Research.* Newbury Park, CA, Sage, ISBN 0071009461.

Rosenthal, R. and Rosnow, R. L. (1991) *Essentials of Behavioral Research: Methods and Data Analysis* . 2nd Edition, New York, McGraw Hill, ISBN: 0070539294.

Rosnow, R. L. and Rosenthal, R. (1996). Computing Contrasts, Effect Sizes, and Counternulls on Other People's Published Data: General Procedures for Research Consumers. *Pyschological Methods,* 1, 331-340.

Ross, P. E. (2005) The Exterminators. *IEEE Spectrum*, 36-41.

Royce, W. W. (1970) *Managing the Development of Large Software Systems: Concepts and Techniques*, Proc, WESCON.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorenson, W. (1991) *Object-Oriented Modeling and Design*. Prentice-Hall, Englewood Cliffs, ISBN: 1-85032-279-1.

Runeson, P. (2003) Using Students as Experiment Subjects – An Analysis on Graduate and Freshman PSP Student Data. *Proceedings of the 8th International Conference on Evaluation and Assessment in Software Engineering*, 95-102.

Satpathy, M., Harrison, R., Snook, C. and Butler, M. (2001) A Comparative Study of Formal and Informal Specifications through an Industrial Case Study. *Proceedings of IEEE/ IFIP Workshop on Formal Specification of Computer Based Systems*.

Scaife, M. and Rogers, Y. (1996) External Cognition: How do Graphical Representations Work?. *International Journal of Human-Computer Studies*, 45, 185-213.

Scanlan, D. A. (1989) Structured Flowcharts Outperform Pseudocode: An Experiment Comparison. *IEEE Software*, 6(5), 28-36.

Schmuller, J. (1999) *Sams teach yourself UML in 24 hours*. Sams Publishing.

Schneider, S. A. (2001) *The B-Method: An Introduction*. Basingstoke, Palgrave, ISBN: 9780333792841.

Seaman, C. B. (1999) Qualitative Methods in Empirical Studies of Software Engineering. *IEEE Transactions on Software Engineering*, 25(4), 557-572.

Seaman, C. B. and Basili, V. (2007) *Empirical Paradigm: Position Paper, Empirical Software Engineering Issues, Lecture Notes in Computer Science 4336*, Springer-Verlag Berlin Heidelberg, 23.

Segal, J., Grinyer, A. and Sharp, H. (2005) The Type of Evidence Produced by Empirical Software Engineers, *Proceedings of the Workshop on Realising Evidence-based Software Engineering*, 1-4.

SEI (2005) *CMMI®, Carnegie Mellon, Software Engineering Institute*. Available online at http://www.sei.cmu.edu/cmmi/ (last accessed July 2006).

Sekerinski, E. (1998) Graphical Design of Reactive Systems, *Proceedings of the 2nd International B Conference, Lecture Notes in Computer Science 1393*, Springer-Verlag.

Senn, S. (1992) Is the Simple Carry-over Model Useful? *Statistics in Medicine*, 11, 715-726.

Senn, S. (2002) *Cross-over Trials in Clinical Research (Statistics in Practice)*, John Wiley & Sons, ISBN: 0471496537.

Shaft, T. M. and Vessey, I. (1995) The Relevance of Application Domain Knowledge: the Case of Computer Program Comprehension. *Information Systems Research*, 6, 286-299.

Shneiderman, B. and Mayer, R. (1979) Syntactic/Semantic Interactions in Programmer Behavior: A Model and Experimental Results. *International Journal of Computer and Information Sciences*, 8, 219-238.

Shneiderman, B. (1980) *Software Psychology: Human Factors in Computer and Information Systems*. Little, Brown and Co., ISBN: 0876268165.

Shneiderman, B. (1998) *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. 3rd Edition, Addison-Wesley, Reading, ISBN: 0-201-69497-2.

Shore, R. (1996) An Object-oriented Approach to B. In Habrias, H. (Ed.), *Putting into Practice Methods and Tools for Information System Design – 1st Conference on the B method*.

Shoval, P. and Frumermann, I. (1994) OO and EER Conceptual Schemas: A Comparison of User Comprehension. *Journal of Database Management*, 5(4), 28-38.

Siegel, S. (1956) *Non-parametric statistics for the behavioral sciences*. New York: McGraw-Hill.

Simon, H. (1974) How big is a chunk?. *Science,* 183, 482-488.

Sjøberg, D. I. K., Hannay, J. E., Hansen, O., Kampenes, V. B., Karahasanović. A., Liborg, N-K. and Rekdal, A. C. (2005) A Survey of Controlled Experiments in Software Engineering. *IEEE Transactions on Software Engineering*, 31(9), 733-753.

Smith, E. E. and Kosslyn, S. M. (2007) *Cognitive Psychology: Mind and Brain*. Upper Saddle River, N.J, Pearson/Prentice Hall, ISBN: 0131825089.

Snook, C. and Harrison, R. (2001) Practitioners' Views on the use of Formal Methods: An Industrial Survey by Structured Interview. *Information and Software Technology*, 43(4), 275-283.

Snook, C. (2002) *Exploring the Barriers to Formal Specification*. PhD Thesis, University of Southampton, UK.

Snook, C. and Sandstrom, K. (2003a) Using UML-B and U2B for Formal Refinement of Digital Components. *Proceedings of Forum on Specification & Design Languages*, Frankfurt.

Snook, C., Tsiopoulos, L. and Walden, M. (2003b) A Case Study in Requirement Analysis of Control Systems using UML and B. *Proceedings of the International Workshop on Refinement of Critical Systems: Methods, Tools and Experience*, Finland.

Snook, C., Butler, M. and Oliver, I. (2003) *Towards a UML profile for UML-B*. Technical Report DSSE-TR-2003-3, Electronics and Computer Science, University of Southampton.

Snook, C., Oliver, I. and Butler, M. (2004a) The UML-B Profile for Formal Systems Modelling in UML. In Mermet, J., ed., *UML-B Specification for Proven Embedded Systems Design*, Springer, Chapter 5.

Snook, C. and Butler, M. (2004b) U2B - A Tool for Translating UML-B Models into B. In Mermet, J., ed., *UML-B Specification for Proven Embedded Systems Design*, Springer, Chapter 6.

Snook, C. and Harrison, R. (2004c) Experimental Comparison of the Comprehensibility of a Z Specification and its Implementation. *Information and Software Technology*, 46, 955-971.

Snook, C., Butler, M., Edmunds, A. and Johnson, I. (2004d) Rigorous Development of Reusable, Domain-Specific Components for Complex Applications. *Proceedings of the 3rd International Workshop on Critical Systems Development with UML*, 115-129.

Snook, C. and Butler, M. (2006) UML-B: Formal Modelling and Design Aided by UML. *ACM Transactions on Software Engineering and Methodology*, 15(1), 92-122.

Solingen, R. and Berghout, E. (1999) *The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development*. McGraw-Hill Publishers, ISBN: 0077095537.

Soloway, E., Ehrlich, K., Bonar, J. and Greenspan, J. (1982) What do Novices Know about Programming? In Badre, A. and Shneiderman, B., (Eds.), *Directions in Human-Computer Interaction*, Albex, Norwood NJ, 27-54.

Sommerville, I. (2001) *Software Engineering*. 6th Edition, Addision-Wesley, ISBN: 020139815.

Spivey, J. M. (1992) *The Z Notation: A Reference Manual*. 2nd Edition, Prentice-Hall, Englewoodcliffs, NJ.

Springer (2007) *SpringerLink*. Available online at http://springerlink.metapress.com (last accessed March 2008).

Stenning, K. and Oberlander J. (1995) A Cognitive Theory of Graphical and Linguistic Reasoning: Logic and Implementation. *Cognitive Science*, 19, 97-140.

Steria, Aix-en-Provence, France (1996) *Atelier B*, *User and Reference Manuals*. Available online at http://www.atelierb.societe.com/index_uk.html (last accessed July 2006).

Strauss, A. L. and Corbin, J. (1998) *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, 2nd Edition, Thousand Oaks, California, ISBN: 0803959397.

Sweller, J., Chandler, P., Tierney, P. and Cooper, M. (1990) Cognitive Load as a Factor in the Structuring of Technical Material. *Journal of Experimental Psychology: General,* 119*,* 176-192.

Sweller, J. (1993) Some Cognitive Processes and their Consequences for the Organisation and Presentation of Information. *Australian Journal of Psychology*, 45*,* 1-8.

Sweller, J. and Chandler, P. (1994) Why Some Material is Difficult to Learn. *Cognition and Instruction*, 12(3), 185-233.

Sweller, J. (1999) *Instructional Design in Technical Areas*. Australian Council for Educational, ISBN 0-86431-312-8.

Tashakkori, A. and Teddlie, C. (1998) *Mixed Methodology: Combining qualitative and quantitative approaches*. Thousand, CA, Sage, ISBN: 0-7619-0070-5.

The Standish Group (1998) *The CHAOS Report*. Available online at http://www.standishgroup.com (last accessed July 2006).

The Standish Group (2001) *Extreme CHAOS*. Available online at http://www.standishgroup.com (last accessed July 2006).

Tichy, W. F. (1995) Experimental Evaluation in Computer Science: A Quantitative Study. *Journal of Systems and Software*, 28(1), 9-18.

Tichy, W. F. (1998) Should Computer Scientists Experiment More?. *IEEE Computer*, 31(5), 32-40.

Tiller, D. (1991) Experimental Design and Analysis. In Fenton, N., (Eds.), *Software Metrics – A Rigorous Approach*, Chapman and Hall.

Tobii (2007) *World Leader in Eye Tracking and Eye Control*. Available online at http://www.tobii.com/corporate/start.aspx (last accessed Dec 2007).

Torchiano, M. (2004) Empirical Assessment of UML Static Object Diagrams. *Proceedings of the 12th IEEE International Workshop on Program Comprehension*, 226-230.

Toval, A., Requena, A. and Alemán, J. L. (2003) Emerging OCL Tools. *Software and System Modeling*, 2(4), 248-261.

Triffitt, E. and Khazaei, B. (2002) A Study of Usability of Z Formalism Based on Cognitive Dimensions. In Kuljis, J., Baldwin, L. and Scoble, R., (Eds.), *Proceedings of the 14th Annual Meeting of the Psychology of Programming Interest Group*, 15-28.

Tukiainen, M. (2001) Evaluation of the Cognitive Dimensions Questionnaire and Some Thoughts about the Cognitive Dimensions of Spreadsheet Calculation. In Kadoda, G., (Eds.), *Proceedings of the 13th Annual Meeting of the Psychology of Programming Interest Group*, 291-301.

University of Southampton (UoS) (2007) *Ethics Policy*. Available online at http://www.soton.ac.uk/research/rso/policies/ethics.html (last accessed July 2007).

University of Surrey (UoS) (2007) *CS389 Formal Software Development*. Available online at https://sits.surrey.ac.uk/live/ipo/COM3008-0002.htm (last accessed July 2007).

USE (2006) *UML-based Specification Environment*. Available online at http://www.db.informatik.uni-bremen.de/projects/USE/ (last accessed Jan 2006).

Van Lamsweerde, A. (2000) Formal Specification: A Roadmap. *Proceedings of the Conference on The Future of Software Engineering Track*, ACM Press, Los Angeles CA, 147-159.

Vaziri, M. and Jackson, D. (1999) *Some Shortcomings of OCL, the Object Constraint Language of UML*, Response to Object Management Group's Request for Information on UML 2.0. Available online at http://www.research.ibm.com/people/m/mvaziri/papers/omg.pdf (last accessed Jan 2008).

Vessey, I. and Weber, R. (1986) Structured Tools and Conditional Logic: An Empirical Investigation. *Communications of the ACM*, 29(1), 48-57.

Vessey, I. (1991) Cognitive Fit: A Theory-Based Analysis of the Graphs Versus Tables Literature. *Decision Sciences*, 22(2), 219-240.

Vessey, I. and Conger, S. (1994) Requirements Specification: Learning Object, Process, and Data Methodologies. *Communications of the ACM*, 37(5), 102–113.

Von Mayrhauser, A. and Vans, A. M. (1995) Program Comprehension during Software Maintenance and Evolution. *Computer*, 28, 44-55.

Von Mayrhauser, A. and Vans, A. M. (1996). Identification of Dynamic Comprehension Processes during Large Scale Maintenance. *IEEE Transactions on Software Engineering*, 22, 424-437.

Von Mayrhauser, A. and Vans, A. M. (1997) Program Understanding Behavior during Debugging of Large Scale Software. In Wiedenbeck, S. and Scholtz, J.C. (Eds.), *Empirical Studies of Programmers: 7th Workshop*, Ablex, Norwood NJ, 157-179.

Wand, Y. and Weber, R. (1993) On the Ontological Expressiveness of Information Systems Analysis and Design Grammars. *Journal of Information Systems*, 3, 217–237.

Wand, Y. and Weber, R.A. (2002) Research Commentary: Information Systems and Conceptual Modeling—A Research Agenda. *Information Systems Research*, 13(4), 363–376.

Wang, S. (1996) Two MIS Analysis Methods: An experimental Comparison. *Journal of Education of Business*, 7(3), 136–141.

Warmer, J. and Kleppe, A. (2003) *The Object Constraint Language: Getting Your Models Ready for MDA*. 2nd Edition, Addison-Wesley, Massachusetts, ISBN: 0321179366.

Weber, R. (2003) Conceptual Modelling and Ontology: Possibilities and Pitfalls. *Journal of Database Management*, 14(3), 1–20.

Westbrook, L. (1994) Qualitative Research Methods: A Review of Major Stages, Data Analysis Techniques, and Quality Controls. *Library and Information Science Research,* 16, 241–245.

Whittaker, J. A. and Voas, J. M. (2003) 50 Years of Software: Key Principles for Quality, *Software Quality Management*, 3(1), 20-23.

Wilcoxon, F. (1945) Individual Comparisons by Ranking Methods. *Biometrics*, 1, 80-83.

Wittrock, M. C. (1989) Generative Processes of Comprehension. *Educational Psychologist,* 24, 345-376.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., C., Regnell, B. and Wesslén, A. (2000) *Experimentation in Software Engineering*. Boston: Kluwer Academic Publishers, ISBN: 978-0-7923-8682-7.

Wohlin, C., Höst, M. and Henningsson, K. (2003) Empirical Research Methods in Software Engineering, In Conradi, R. and Wang, A. I. (Eds.)*Empirical Methods and Studies in Software Engineering*, *Lecture Notes in Computer Science 2765*, Springer-Verlag Berlin Heidelberg, 7-23.

Wood, M., Daly, J., Miller, J. and Roper, M. (1999) Multi-method Research: An Empirical Investigation of Object-oriented Technology. *Journal of Systems and Software*, 48(1), 13-26.

Yadav, S., Bravoco, R., Chatfield, A and Rajkumar, T. (1988) Comparison of Analysis Techniques for Information Requirements Determination. *Communications of the ACM*, 31(9), 1090–1097.

Yin, R. K. (2003) *Case Study Research: Design and Methods*. 3rd Edition, Sage Publications, California, ISBN: 076192552.
Yourdan, E. (1989) *Modern Structured Analysis*, Englewood Cliffs, Prentice-Hall, ISBN: 0-13-598624-9.

Zannier, C., Melnik, G. and Maurer, F. (2006) On the Success of Empirical Studies in the

International Conference on Software Engineering. *Proceedings of the 28th International Conference on Software Engineering*, 341-350.

Zave, P. (1990) A Comparison of the Major Approaches to Software Specification and Design. In Thayer, R.H. and Dorfman, M. (Eds.), *System and Software Requirements Engineering*, IEEE Computer Society Press.

Zhang, J. (1997) The Nature of External Representations in Problem Solving. *Cognitive Science*, 21, 179-217.

Zelkowitz, M. V. and Wallace, D. (1997) Experimental Validation in Software Engineering. *Information and Software Technology*, 39(11), 735-743.

Zelkowitz, M. V. and Wallace, D. R. (1998) Experimental Models for Validating Technology. *IEEE Computer*, 31(5), 23-31.

Zelkowitz, M. V. (2007) Techniques for Empirical Validation. In Basili, V. et al. (Eds.), *Empirical Software Engineering Issues*, *Lecture Notes in Computer Science 4336*, Springer-Verlag Berlin Heidelberg, 4-9.

Zimmerman, M.K., Lundqvist, K. and Leveson, N. G. (2002) Investigating the Readability of State-based Formal Requirements Specification Languages. *Proceedings of the 24th International Conference on Software Engineering*, ACM Press, Los Angeles CA, 33-43.