HUMAN TECHNOLOGY

An Interdisciplinary Journal on Humans in ICT Environments ISSN: 1795-6889

www.humantechnology.jyu.fi Volume 4 (1), May 2008, 26–46

USABILITY ASSESSMENT OF A UML-BASED FORMAL MODELING METHOD USING A COGNITIVE DIMENSIONS FRAMEWORK

Rozilawati Razali
Dependable Systems and Software Engineering, School of Electronics and Computer Science, University of Southampton, UK

Colin Snook
Dependable Systems and Software Engineering, School of Electronics and Computer Science, University of Southampton, UK

Michael Poppleton
Dependable Systems and Software Engineering, School of Electronics and Computer Science, University of Southampton, UK

Paul Garratt
Dependable Systems and Software Engineering, School of Electronics and Computer Science, University of Southampton, UK

Abstract: *Conceptual models communicate the important aspects of a problem domain to stakeholders. The quality of the models is highly dependent on the usability of the modeling method used. This paper presents a survey conducted on a method that integrates the use of a semiformal notation, namely the Unified Modeling Language (UML) and a formal notation, namely B. The survey assessed the usability of the method by using the grounded theory, the Cognitive Dimensions of Notations (CD) framework, and several criteria suggested by the International Organization for Standardization (ISO). Ten participants responded to the survey. The results suggest that the method is accessible to users when the principles and roles of each notation are obvious and well understood, and when there is strong support from the environment. Supported by the findings, a usability profile based on CD for designing a method that integrates semiformal and formal notations is proposed.*

Keywords: *empirical assessment, semiformal and formal notations, cognitive dimensions (CD), grounded theory, usability.*

INTRODUCTION

Modeling 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 modeling 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. Some examples of notations used in conceptual modeling include semiformal notations such as entity-relationship diagram (ERD; Chen, 1976) and Unified Modeling Language (UML; Object Management Group [OMG], 2008), and formal notations such as Z (Spivey, 1992) and B (Abrial, 1996). In addition, there are also notations that integrate both semiformal and formal, such as UML and Z (Martin, 2003).

Formal notations such as Z and B use mathematical symbols to describe a system. The notations have three components: 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 a model (van Lamsweerde, 2000). Formal notations therefore have the ability to increase a model's precision and consistency, which is necessary especially for critical systems (Hinchey, 2002). However, the notations are regarded as being difficult to comprehend, due to the usage of unfamiliar symbols and underlying rules of interpretation that are not apparent to many practitioners (Carew, Exton, & Buckley, 2005). On the other hand, semiformal notations such as ERD and UML provide abstract graphical representations for illustrating system elements. They are semiformal because, although they possess some formal aspects such as support the iterative refinement process, they cannot be used to verify or predict the vast majority of system characteristics (Alexander, 1996). As a result, an accurate and consistent model cannot be guaranteed. Nonetheless, the notations are perceived as more accessible, since it is easier to visualize the mapping of graphical symbols to the real-world objects they represent (Bauer & Johnson-Laird, 1993).

By integrating formal and semiformal notations, it may be that practitioners can produce a model that is accurate, consistent, and more accessible to them. One possible approach to this integration is to combine the formal notation of B and the semiformal notation of UML. A method called UML-B (Snook & Butler, 2006) is one such product. The rationale of this integration is that B has strong industrial supporting tools, such as Atelier-B (ClearSy Systems Engineering [ClearSy], n.d.) and B-Toolkit (B-Core Limited [B-Core], 2002), and UML has become the de facto standard for system development (Pender, 2003).

This paper presents an investigation into the usability of UML-B. Usability in this context means the understandability/comprehensibility, learnability, operability, and attractiveness of the method. The assessment was conducted by using the grounded theory and a usability evaluation framework, namely the Cognitive Dimensions of Notations (CD; Green, 1989; Green & Petre, 1996), with several usability criteria suggested by the International Organization for Standardization (ISO, 2003, 2004). The following section provides the background of the paper, which includes a brief description of CD and UML-B. Later, the survey is presented. The final section concludes the paper with a summary of the main findings and future work.

BACKGROUND

Cognitive Dimensions

The CD framework 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 & Petre, 1996). Nevertheless, it is also applicable beyond the programming environment. Since its proposal, the CD framework has been used as a basis of usability evaluation for several notations, such as UML (Cox, 2000; Kutar, Britton, & Barker, 2002), C# (Microsoft Corporation [Microsoft], 2008) programming language (Clarke, 2001), spreadsheet application (Tukiainen, 2001), and Z notation and tools (Triffitt & Khazaei, 2002).

The framework is generally seen as a tool that aids the usability evaluation of information-based artifacts (Green & Blackwell, 1998). The aim of the framework is to provide general guidelines that can be used to evaluate the usability and suitability of an artifact for a particular setting. An artifact is analyzed based on a usability profile that contains a CD set. The profile guides the evaluation of the artifact for a particular user activity. The framework distinguishes six main types of user activity (Blackwell & Green, 2003): incrementation, transcription, modification, exploratory design, searching and exploratory understanding. Each of these user activities is supported by a specific usability profile.

Table 1 provides the 14 dimensions in the CD framework, with summarized descriptions. Although the dimensions are conceptually independent, many of the dimensions are pairwise interdependent (Green & Blackwell, 1998). This means although any given pair can be treated as independent, a change in one dimension usually requires a change in some other dimension. For example, 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 a matter of making compromises or tradeoffs in artifact designs.

Table 1. The CD Framework (drawn from Green, 1989).

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

In essence, CD provides a framework for assessing the usability of building and modifying information structures. Because usability depends on the structure of the notation and the supporting tools provided by the environment, the dimensions are indeed applicable to the whole system.

UML-B

UML-B (Snook & Butler, 2006) is a graphical formal modeling notation and method based on UML (OMG, 2008) and B (Abrial, 1996). It uses UML's *Class* and *Statechart* diagrams as the graphical representation of its model. The Class diagram shows the structure and the relationships between system entities. The Statechart diagrams are attached to classes to describe their behavior. A notation, μB (micro B) that is based on B notation, is used for textual constraints and actions for the diagrams. μB has an object-oriented style dot notation that is used to show ownership of entities, namely attributes and operations by classes. The modeling environment of UML-B includes Rational Rose (IBM Software [IBM], n.d.) and a translator called U2B (Snook & Butler, 2006). Rational Rose provides the environment for the UML-B model development while U2B is a tool that translates a UML-B model to a B model so that it can be verified by B tools, such as Atelier-B (ClearSy, n.d.) and B-Toolkit (B-Core, 2002).

Figures 1 and 2 illustrate examples of a Class diagram and a Statechart diagram of a UML-B model, respectively. The Class diagram shows the entities and relationships involved in an Auction System. Two main classes, namely *USER* and *AUCTION*, are connected through *seller* and *highest_bidder* relationships. The Statechart diagram shows the states and transitions (operations) of the AUCTION class with the respective textual constraints specified using μB.

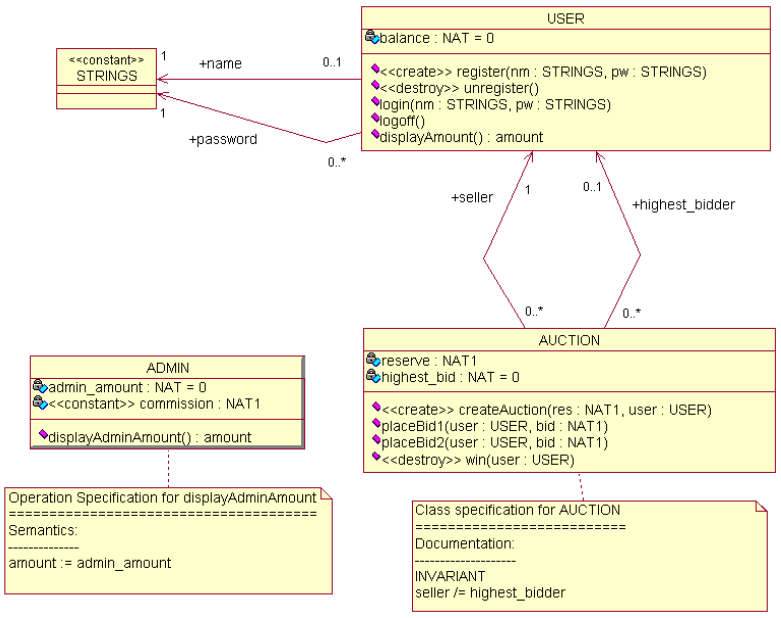

Figure 1. An example of a Class diagram of UML-B.

Figure 2. An example of a Statechart diagram of UML-B.

The comprehensibility of the notation used in a UML-B model has been assessed in previous work (Razali, Snook, Poppleton, Garratt, & Walters, 2007). The assessment was conducted as a controlled experiment that compared a UML-B model and a B model for model interpretation task. The measure of interest used in the experiment was efficiency in performing the task, that is, accuracy over time. The results suggest with 95% confidence that a UML-B model could be up to 16% (overall comprehension) and 50% (comprehension for modification task) easier to understand than the corresponding B model. The subjects commented that the UML-B model made it easier and quicker to understand the scenario and the relationships between operations; easy to develop, especially on computers; and more logical to developers. Nevertheless, the model was said to be useful only with good tool support. The UML-B model was also regarded as being quite "messy," since the information was scattered around the Class and Statechart diagrams.

SURVEY

The controlled experiment described briefly in the previous section 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 modeling. Neither could they determine whether or not the notation suits the developers' common needs and expectations.

The following subsections present a further survey conducted on UML-B. The survey assessed the usability of the notation used in UML-B from developers' perspectives, especially from the point of view of users who have only recently started to use it. Since

usability depends on the notation and its environment, the evaluation included the tools that accompany the method, namely Rational Rose (IBM, n.d.) and U2B (Snook & Butler, 2006), whenever appropriate.

Objectives and Methods

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. Since little is known about the UML-B 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 analyzed.

Many different approaches to qualitative data are employed in the social sciences (Cassell & Symon, 1994; Denzin & Lincoln, 1994; Westbrook, 1994). We adopted one approach, namely the grounded theory (Glaser & Strauss, 1967; Strauss & Corbin, 1998). There are two variations in the approach, which are based on different directions taken by its originators, namely Glaser (1992) and Strauss and Corbin (1998). This survey employed Strauss' 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 amid the masses of data. In a qualitative study, the research question should be broad and open-ended.

The theory in the grounded theory approach is derived from data, systematically gathered and analyzed through the process. This approach was chosen because, unlike the controlled experiment conducted previously, this survey was not based on any specific theory. The grounded theory approach allows the study to be initiated without a preconceived theory in mind: The researcher can start with a phenomenon and allow the theory to emerge from the collected data. Because the theory is drawn from data, it is likely to offer insight, enhance understanding, and provide a meaningful guide to action (Strauss & Corbin, 1998). It is believed that the theory generated from this approach is more likely to resemble the reality, as compared to theory derived by merging concepts based on how one thinks things ought to work.

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

One of the subjective comments obtained from the earlier controlled experiment was that UML-B was seen as easy to develop, particularly on computers. The method also was deemed to be useful only with good tool support. These hypotheses were given by subjects who dealt with the already-developed UML-B model, not the process of modeling. This could suggest, therefore, that the hypotheses might not be true from developers' perspectives

for modeling purposes. As a result, the survey included these hypotheses in its investigation of the phenomenon through the following broad research questions:

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

Materials

The survey instrument was developed based on the ideas proposed in the CD usability framework (Green, 1989). The framework was adopted because it captures a significant number of psychology and human–computer interaction (HCI) aspects that focus particularly on the notational design. The framework comprises 14 dimensions (see Table 1), which acted as the response variables in the survey.

The questions for the survey were constructed by following the proposed CD questionnaire (Blackwell & Green, 2000). The advantage of using a standard instrumentation, as proposed by the CD questionnaire, is that it has been assessed for validity and reliability by the authors. The CD framework is widely used by other researchers investigating the usability of notations, such as UML diagrams (Kutar et al., 2002) and Z (Triffitt & Khazaei, 2002), and so it 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 artifacts, rather than presenting descriptions specialized to a specific system under consideration. The questionnaire was therefore tailored and modified slightly to reflect the characteristics of UML-B. Moreover, the questions for the survey were designed to include a set of answers using an ordinal scale together with the open-ended questions. This approach allowed the survey to obtain some quantitative measures rather than exclusively qualitative measures.

In addition to the CD framework, the questions on the survey were also constructed based on the usability criteria proposed by the International Organization for Standardization (ISO, 2003, 2004): understandability, learnability, operability, and attractiveness. There were 20 questions on the survey: 14 reflecting the dimensions of the CD framework, 5 representing the ISO's usability criteria, and 1 designed to gather suggestions for improvement. The 14 questions on CD were also mapped to at least one usability criterion of ISO. The mapping was based on the definition stated in the standard. The questions on 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 & Pfleeger, 2002).

The questions used an ordinal scale that provided the respondents with five potential levels of agreement, from *–2* (*very difficult)* to *2* (*very easy).* An uneven number of levels were used because, by allowing for a neutral opinion, uneven numbers contribute to the achievement of better results (Bonissone, 1982). In addition to the selection on the scale, justification for the answer given was also required through open-ended questions, 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. There were also questions that required an answer of Yes, No or Not sure.

The survey questions and raw data can be found in Razali (2007). As an overview of the questions, Figure 3 provides some examples of the survey questions. The first question concerns the visibility and juxtaposability dimension, which also relates to the operability/attractiveness criteria of the ISO. The second question involves the hard mental operations dimension that also implies the ISO's understandability/learnability criteria.

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 & Green, 2003). In particular, it addresses 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, in which the users employed UML-B (notation and environment) to design a conceptual model. The survey questions and analysis therefore were tailored towards 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 or unnecessary questions as well as to identify any ambiguous questions and instructions.

Participants

Ten participants responded to the survey. They were master's students of a software engineering program at the University of Southampton, who registered for the Critical Systems[1] course in spring 2006. They were chosen due to their potential contribution towards the development of usability theory for integrated methods such as UML-B. Specifically, they were selected because they received formal training on B (9 hours) and UML-B (1 hour) during the course. They also had completed courses on the object-oriented technology and formal methods of developing at some points in their studies. Basic knowledge of those aspects is necessary to develop a UML-B model. Moreover, the participants had some practical experience in using UML-B and its tools before participating in the survey. In particular, they used the method to develop a model of a system as part of their coursework towards the end of the Critical Systems course.

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 modeling your UML-B model?

No Not Sure Yes

If Yes, what are they? If No or Not Sure, why?

Figure 3. Examples of the survey questions from Razali (2007).

The survey adhered to the university's ethical policies and guidance for conducting research involving human participants. 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 in addition to providing a space for reflection on their learning prior to their course examination.

The subjects were in the final semester of their master's program. They therefore had a reasonable amount of experience and knowledge in software development. Some of them had some professional work experience in this area. They are the next generation of professionals, thus they represented closely the population under study: software developers who are new users of the UML-B method.

Results and Analysis

The survey adopted the grounded theory approach for the data analysis. In addition to capturing the informants' experiences of using UML-B, the survey aimed to formulate tentative theories on the 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 number of coding and analysis processes. The first one applied was *open coding* where the responses were examined for objects of interest based on the stated research questions. The technique used was *microanalysis* (Strauss & Corbin, 1998). The analysis focused on identifying 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 ones specific to any individual cases. For example, issues of using Rational Rose (IBM, n.d.) and running U2B (Snook & Butler, 2006) were conceptualized 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, an 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 recognized as one of the categories. On the other hand, it is necessary to know what aspect of the notations and their tools was easy or 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. The use of CD framework and ISO's usability criteria that shaped the dimensions of usability investigation facilitated the identification of the categories and properties.

The following paragraphs list the categories and elaborate their properties. The properties (reasoning based on CD and ISO usability criteria) that support the statements are stated in the parentheses in the paragraphs. The properties were grouped into categories based on the respondents' qualitative answers and data on the ordinal scales (for details, see Razali, 2007).

Category 1: Model Structure and Organization. The UML portion of UML-B allows the system properties and behaviors 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 modeling 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 behaviors of the systems are more precisely specified. Formal semantics in the form of B syntax are added at different parts of the diagrams so that the diagrams and semantics 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 behaviors are placed at the Statechart diagram level. Despite being scattered throughout 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 for the users. They need to switch to 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 to become 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 modeling stage they like (Property: progressive evaluation dimension). However, having to transform the model, particularly while formulating and synthesizing ideas, has been regarded as a "noise." In addition, model transformation at early stages, where many aspects have yet to be carefully thought through, will generate error messages in B tools. And starting modeling 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 caused not only by the scattered information but also the display of multiple windows simultaneously. 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 organized and presented has an effect (Chandler & Sweller, 1992). When the related information is separated on the page or screen, users have to use cognitive resources to search and integrate it. Users are less able to hold the separated information in working

memory simultaneously, especially if the information has a high intrinsic cognitive load (Sweller & Chandler, 1994). In general, a formal notation such as B syntax is high in intrinsic cognitive load because it involves concurrent interactions between its syntactical and semantic characteristics.

Because 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 modeling tool allows more convenient and less distracting switching to and viewing different parts of the model.

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, several problems in user-friendliness were 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 into 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 modeling a B model from scratch. At the very least, it provides basic structures for the B model, which the users could extend further by adding more details. However, in order to keep the U2B simple, it does not contain a verification feature; the user would need to return to the B tools to achieve verification. As a result, no matter how simple to use, U2B, or even Rational Rose, does not support any type of checking. This means users have to transform the UML-B model to a B model and run it in B tools each time they change an idea, even if it involves only a minor change. Otherwise, there is no way to be sure whether or not the change is acceptable. The generated B model will contain numerous types of errors from the simplest to the most complex, which can only be recognized during model verification using B tools. Because of this reason, users feel that the method is less supportive for experimenting with ideas (Property: provisionality dimension). Users would benefit from having some simple checking abilities, such as unused variables and typing errors of B syntax at the modeling and transformation levels. This could act as the frontline checking to eliminate minor 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 for the users. Because the tool currently lacks 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 modeling environment should be created so that users do not have to perform several individual and intricate steps during modeling.

Category 3: Learnability of Notations and Tools. The successful use of UML-B 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. From the results of the survey (Razali, 2007), 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, a level of mental burden still occurs during the process (Property: hard mental operations dimension). Having to think, integrate, and harmonize two styles of modeling 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, behaviors 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. Users have to understand these transformations and why they are accomplished in such ways (Property: Learnability and Utility of U2B; hidden dependencies dimension), since it affects the way they should do the modeling (Property: closeness of mapping dimension). Moreover, learning of how to do modeling in Rational Rose is also required (Property: Learnability of UML-B).

Modeling 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 modeling 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 modeling (Property: Operability and Attractiveness of UML-B).

On the other hand, specifying the UML diagrams with the correct formal semantics is perceived as difficult 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 employed 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 of the practical aspects 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)

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 of 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 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 modeling, the flexibility somehow has made the role of the semantics in the Statechart diagram, or even the Statechart diagram itself, unclear to some users (Property: role expressiveness-diagram dimension; role expressiveness-syntax dimension). The users seem to prefer specifying the full semantics in the Class diagram, since it is more obvious and straightforward. Such a process 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. Because the only end product that actually matters is the transformed B model, 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 of the roles and responsibilities of each part of the diagrams and semantics should be stated succinctly in the documentation, which is currently lacking in the method (Property: Usefulness of Documentation). It may be better if some principles and controls can be placed on how a UML-B model should be modeled. Although it may reduce the flexibility in modeling, it could 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 often have no idea how to start and pursue the modeling. Furthermore, the transformation of formal semantics from the Statechart diagram to a B model could be smoothed further so that no unnecessary complication is introduced to users.

Discussion

The data from the survey suggest that UML-B is appealing to users who opt into B modeling while yet prefer working with 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 modeling environment alleviates the difficulty of developing a formal model from scratch by stimulating the formulation of ideas through the use of visual objects at the abstraction level. On the other hand, users are faced with the challenge of having to grasp the underlying principles of each unique 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. 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.

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

> **Theory 1:** The integration of semiformal 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 (Categories 3 and 4).
>
> **Theory 2:** The integration of semiformal 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 (Categories 1, 2, and 3).

Unlike the other categories, Category 1: Model Structure and Organization 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 within different environments.

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 semiformal and formal notations, and the motivation behind the integration. The individual notations (semiformal and formal) have their own strengths and weaknesses, which are enhanced through the integration effort. In addition, the design goals were 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 implement such methods to create a new model, and modification, where users use the methods to make changes and enhancements to an existing model.

Table 2 illustrates the recommended CD profile for designing methods that combine semiformal and formal notations. The profile proposes the desired level for each dimension that integrated methods and their notations (a combination of semiformal and formal) should aim to achieve after the integration. 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 a Modification activity since it provides users with additional informal information. It thus may be needed (High) to improve the model comprehensibility, especially for formal (mathematical) models. However, secondary notation may cause exploratory design activity to be a bit cumbersome, because users are obliged to provide informal information about the elements in the model in addition to 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 difficult because the changes are also required for the additional information. 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

Table 2. Proposed CD Profile for Designing Integrated Methods of Semiformal and Formal Notations.

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	
Provisionality	High
Role-expressiveness	High*
Secondary notation	Moderate (instead of High)
Viscosity	Low
Visibility/Juxtaposibility	High

Note: High means to increase; Low means to reduce; Moderate suggests a possible trade-off among dimensions;
*Semiformal notations support formal notations to achieve the desired level (otherwise, the level will be opposite);
**Formal notations support semiformal notations to achieve the desired level (otherwise, the level will be opposite).

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, or fuller. 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. A single asterisk in Table 2 indicates a dimension that affects formal notations, which semiformal notations help to reduce the effect. On the other hand, two asterisks denote a dimension that semiformal 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 difficulties. The use of intuitive graphical symbols in semiformal notations with formal notations often reduces the effect. Similarly, semiformal notations in general lack mechanisms for a systematic progressive evaluation, which formal notations can normally offer. Without such interplay between the two types of notations, the integration is not worth the effort. 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 cooperate to achieve the desired level for dimensions other than those 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. Integrating the graphical symbols of semiformal notations with formal notations may alleviate the effect, since the grouping of elements becomes more apparent (Low).

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

Consistency: The formality in formal notations enforces a consistency that semiformal notations solely could not assure (Low). Semiformal notations together with formal notations could enable a consistent graphical formal model to be developed (High).

Diffuseness: The textual aspect of formal notations that is similar to natural language may cause a description to be fuller. In contrast, the graphical symbols in semiformal notations could normally carry meanings in simpler forms. The combination of textual and graphical symbols may enable the description to be short and precise (Low or Moderate).

Error-proneness: The unfamiliar mathematical symbols in formal notations frequently induce mistakes (High). The accessibility of graphical symbols in semiformal notations may reduce the tendency of making errors (Low).

Premature commitment: Formal notations normally require users to look ahead in order to obtain the right abstractions (High). Incorporating the graphical symbols of semiformal notations into formal notations may reduce the effect, since they permit the visualization of possible interacting entities (Low).

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 semiformal notations are mainly intuitive. By combining the graphical symbols together with the mathematical symbols, users may be helped to grasp the roles of the latter (High).

The remaining dimensions without a single or double asterisk in Table 2 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 in achieving the desired levels for these dimensions. This environment includes the structure of the model and the tools that support the notations. This claim is worth investigating in future.

The tentative theories and the proposed CD profile may not be conclusive, and they should 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 provide a meaningful guide to better design.

Validity

Threats to validity are influences that may limit the ability to draw conclusions from the data. The following paragraphs discuss some threats of this survey.

Selection of Respondents. The respondents were students in the university where the research was conducted. Therefore, their answers might have been biased (positively or

negatively). On the other hand, the respondents were considered the most appropriate candidates for this study because they have been trained on B and UML-B. This knowledge is necessary for using UML-B. In fact, the participants also had some experience in using UML-B and thus were able to contribute more fully 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.

Students as Respondents. The respondents of this survey were students. They may have not represented software developers, since they are less experienced. However, the respondents were in the final semester of their master's program and had a reasonable amount of experience and knowledge of software development. Half of the students had some professional working experience. Thus they were seen as valid respondents for the survey as new users with developer's experience.

Sample Size and Response Rate. The survey questionnaire was distributed to all 14 master's students of software engineering at the University of Southampton who registered for the Critical Systems course in spring 2006. Thirteen students responded to the survey. Due to a technical problem, only 10 responses were considered for analysis. Although the number was quite small, a response rate of 70% was considered appropriate for an initial attempt. Moreover, as a qualitative study, the quality of the data is the focus, rather than strictly 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.

Non-committal Responses. Using an uneven number of levels for the ordinal scale leaves open the possibility of noncommittal responses, with the medians representing "neither –nor" or "not sure." Although such incidents could be seen in the data, they did not happen often and no pattern was detected in either the questions or by respondents.

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

Analysis Process. The grounded theory approach encourages the gathering of further data after analyzing the first gathered data. In fact, data collection and analysis should be repeated several times so that more incidents are captured and validated until the theory saturates (Strauss & Corbin, 1998). Due to time and resources constraints, the data collection and analysis for the survey were conducted only once, and the findings presented here reflect one set of data. However, the survey will be repeated in the future.

Nature of Study. Surveys and qualitative measures by their nature are retrospective. Therefore, there was a risk that the respondents reported 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 completed the modeling task could have reduced this threat, because the respondents would have had a clearer memory of what they found during the task. The respondents submitted the questionnaire together with their completed models at the end of the course.

Heterogeneity of Respondents. The respondents might have different abilities and experiences. Thus, there was a risk that the results might have been affected by individual differences. As a qualitative study, the variation however could provide richer data for the analysis.

Familiarity of Respondents. The respondents were taught formally on B for about 9 hours and on UML-B for 1 hour. They were then required to complete a modeling task using UML-B within a month period. 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' perspectives. Therefore, the allocated time frame and training were seen as adequate and realistic for the purpose of this research. The results may also have been influenced by the respondents' knowledge of UML obtained from their previous working experience and studies, which varied considerably.

CONCLUSION

This paper has presented a survey conducted on a method that integrates the use of semiformal and formal notations, namely UML-B. The survey assessed the usability of the notation used in the method and its modeling environment by using the CD framework with several usability criteria suggested by the ISO. The data analysis was conducted using the grounded theory approach. The findings indicated that the dual characteristics of the method bring to users several implications, both positive and negative. Combining semiformal 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 to the designers the loads of two individual notations, which are actually quite different in many ways. 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 modeling process but also resources for learning the method. Based on the findings, we proposed a usability profile based on CD for designing integrated methods such as UML-B.

Some of the findings of the investigation are now being fed into the next generation of UML-B development[2]. The findings of the survey can be improved further by extending the survey to a large number of users. This will help enhance the current understanding of the method and discovering other factors that might affect its use. The tentative theories and the proposed CD profile of integrated methods (combined semiformal and formal notations) discussed in this paper 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.

ENDNOTES

1. Electronics Computer Science (ECS), COMP3011 Critical Systems, http://www.ecs.soton.ac.uk/syllabus/COMP3011.html
2. EU Framework VI project: Rigorous Open Development Environment for Complex Systems (RODIN) http://rodin.cs.ncl.ac.uk/

REFERENCES

Abrial, J. R. (1996). *The B-Method: Assigning programs to meanings*. Cambridge, UK: Cambridge University Press.

Alexander, P. (1996). Best of both worlds (formal and semi-formal software engineering). *IEEE Potentials*, *14*, 29–32.

Bauer, M., & Johnson-Laird, P. (1993). How diagrams can improve reasoning. *Psychological Science*, *4*, 372–378.

B-Core Limited [B-Core]. (2002). The *B-Toolkit*. Retrieved April 18, 2008, from http://www.b-core.com/ONLINEDOC/BToolkit.html

Blackwell, A. F., & Green, T. R. G. (2000). A cognitive dimensions questionnaire optimised for users. In A. F. Blackwell & E. Bilotta (Eds.), *Proceedings of the 12th Workshop of the Psychology of Programming Interest Group* (PPIG '00; pp. 137–154). Cosenza, Italy: Memoria.

Blackwell, A., & Green. T. (2003). Notational systems: The cognitive dimensions of notations framework. In J. M. Carroll (Ed.), *HCI models, theories and frameworks: Toward a multidisciplinary science* (pp. 103–134). San Francisco: Morgan Kaufmann.

Bonissone, P. (1982). A fuzzy sets based linguistic approach: Theory and application. In M. Gupta & E. Sanchez (Eds.), *Approximate reasoning in decision analysis* (pp. 329–339). New York: North-Holland Publishing Company.

Carew, D., Exton, C., & Buckley, J. (2005). An empirical investigation of the comprehensibility of requirements specifications. In G. Kadoda (Ed.), *Proceedings of the 4th International Symposium on Empirical Software Engineering* (ISESE '05; pp. 256–266). Noosa Heads, Australia: IEEE Computer Society.

Cassell, C., & Symon, G. (1994). *Qualitative methods in organizational research.* Thousand Oaks, CA, USA: Sage.

Chandler, P., & 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*, 9–37.

Clarke, S. (2001). Evaluating a new programming language. In G. Kadoda (Ed.), *Proceedings of the 13th Workshop of the Psychology of Programming Interest Group* (PPIG '01; pp. 275–289). Bournemouth, UK: Bournemouth University.

ClearSy Systems Engineering [ClearSy]. (n.d.). *Atelier B, the industrial tool to efficiently deploy the B Method.* Retrieved April 18, 2008, from http://www.atelierb.eu/index_en.html

Cox, K. (2000). Cognitive dimensions of use cases: Feedback from a student questionnaire. In A. F. Blackwell & E. Bilotta (Eds.), *Proceedings of the 12th Workshop of the Psychology of Programming Interest Group* (PPIG '00; pp. 99–122). Cosenza, Italy: Memoria.

Denzin, N., & Lincoln, Y. (1994). *Handbook of qualitative research.* Thousand Oaks, CA, USA: Sage.

Glaser, B. (1992). *Basics of grounded theory analysis: Emergence vs. forcing*. Mill Valley, CA, USA: Sociology Press.

Glaser, B. G., & Strauss, A. L. (1967). *The discovery of grounded theory: Strategies for qualitative research*. London, UK: Weidenfeld and Nicolson.

Green, T. R. G. (1989). Cognitive dimensions of notations. In A. Sutcliffe & L. Macaulay (Eds.), *People and computers V* (pp. 443–460). Cambridge, UK: Cambridge University Press.

Green, T. R. G., & Blackwell, A. F. (1998, September). Design for usability using cognitive dimensions. Tutorial session at the *British Computer Society Conference on Human Computer Interaction* (BCS-HCI '98). Sheffield, UK.

Green, T. R. G., & Petre, M. (1996). Usability analysis of visual programming environments: A cognitive dimensions framework. *Journal of Visual Languages and Computing*, *7*, 131–174.

Hinchey, M. G. (2002). Confessions of a formal methodist. In P. A. Lindsay (Ed.), *Proceedings of the 7th Australian Workshop on Safety-Related Programmable Systems* (SCS '02; pp. 17–20). Adelaide, Australia: Australian Computer Society.

IBM Software [IBM]. (n.d.). *Rational Rose*. Retrieved April 18, 2008, from http://www-306.ibm.com/software/awdtools/developer/rose/index.html

International Organization for Standardization [ISO]. (2003, July). *Software engineering, product quality—Part 3: Internal metrics* (Standard No. 9126-3). Geneva, Switzerland: ISO.

International Organization for Standardization [ISO]. (2004, March). *Software engineering, product quality—Part 4: Quality in use metrics* (Standards No. 9126-4). Geneva, Switzerland: ISO.

Kitchenham, B. A. & Pfleeger, S. L. (2002). Principles of survey research: Part 3: Constructing a survey instrument. *SIGSOFT Software Engineering Notes, 27*(2), 20–24.

Kutar, M., Britton, C., & Barker, T. (2002). A comparison of empirical study and cognitive dimensions analysis in the evaluation of UML diagrams. In J. Kuljis, L. Baldwin, & R. Scoble (Eds.), *Proceedings of the 14th Workshop of the Psychology of Programming Interest Group* (PPIG '02; pp. 1–14). Brunel, UK: Brunel University College.

Martin, S. (2003). The best of both worlds integrating UML with Z for software specifications, *Journal of Computing and Control Engineering, 14,* 8–11.

Microsoft Corporation [Microsoft]. (2008). Visual C# Developer Center. Retrieved April 18, 2008, from http://msdn.microsoft.com/vcsharp/

Object Management Group [OMG]. (2008). *Introduction to OMG's unified modeling language (UML).* Retrieved April 18, 2008, from http://www.omg.org/gettingstarted/what_is_uml.htm

Pender, T. (2003). *UML Bible*. Indianapolis, IN, USA: Wiley.

Razali, R. (2007). *UML-B Survey questionnaires and responses*. (Electronics and Computer Science, University of Southampton Tech. Rep., ID code 13322). Retrieved April 18, 2008, from http://eprints.ecs.soton.ac.uk/13322

Razali, R., Snook, C. F., Poppleton, M. R., Garratt, P. W., & Walters, R. J. (2007). Experimental comparison of the comprehensibility of a UML-based formal specification versus a textual one. In B. Kitchenham, P. Brereton, & M. Turner (Eds.), *Proceedings of the 11th International Conference on Evaluation and Assessment in Software Engineering* (EASE '07; pp. 1–11). Keele, UK: British Computer Society.

Snook, C., & Butler, M. (2006). UML-B: Formal modelling and design aided by UML. *ACM Transactions on Software Engineering and Methodology*, *15*(1), 92–122.

Spivey, J. M. (1992). *The Z notation: A reference manual* (2nd ed.). Englewood Cliffs, NJ, USA: Prentice-Hall.

Strauss, A. L., & Corbin, J. (1998). *Basics of qualitative research: Techniques and procedures for developing grounded theory* (2nd ed.). Thousand Oaks, CA, USA: Sage.

Sweller, J., & Chandler, P. (1994). Why some material is difficult to learn. *Cognition and Instruction, 12,* 185–233.

Triffitt, E., & Khazaei, B. (2002). A study of usability of Z formalism based on cognitive dimensions. In J. Kuljis, L. Baldwin, & R. Scoble (Eds.), *Proceedings of the 14th Workshop of the Psychology of Programming Interest Group* (PPIG '02; pp. 15–28). Brunel, UK: Brunel University College.

Tukiainen, M. (2001). Evaluation of the cognitive dimensions questionnaire and some thoughts about the cognitive dimensions of spreadsheet calculation. In G. Kadoda (Ed.), *Proceedings of the 13th Workshop of the Psychology of Programming Interest Group* (PPIG '01; pp. 291–301). Bournemouth, UK: Bournemouth University.

van Lamsweerde, A. (2000). Formal specification: A roadmap. In *Proceedings of the Conference on the Future of Software Engineering* (pp. 147–159). New York: ACM Press.

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.

Authors' Note

The authors gratefully acknowledge the COMP3011 (spring 2006) students who participated in this study.

All correspondence should be addressed to:
Rozilawati Razali or Colin Snook
Dependable Systems and Software Engineering Group (DSSE)
School of Electronics and Computer Science (ECS)
University of Southampton
SO17 1BJ United Kingdom
rr04r@ecs.soton.ac.uk or rozila_razali@yahoo.co.uk or cfs@ecs.soton.ac.uk

Human Technology: An Interdisciplinary Journal on Humans in ICT Environments
ISSN 1795-6889
www.humantechnology.jyu.fi