Journal of Computer Science 6 (10): 1189-1198, 2010
ISSN 1549-3636

Usability Requirements of Formal Verification Tools: A Survey

[1]Rozilawati Razali and [2]Paul Garratt
[1]School of Computer Science, Faculty of Information Science and Technology,
University Kebangsaan Malaysia, 43600 UKM Bangi, Selangor, Malaysia
[2]School of Electronics and Computer Science,
Faculty of Engineering, Science and Mathematics, University of Southampton,
SO17 1BJ United Kingdom

Abstract: Problem statement: Formal notations employ mathematical symbols and interpretation to illustrate system elements. The formality imposed by the notations allows the accuracy and consistency of a system model to be confirmed by verification tools. Formal notations on the other hand are difficult to understand and use by most users. As supporting instruments, verification tools are expected to be as usable as possible to overcome this limitation. **Approach:** This study presented a survey conducted on two instances of verification tools that support a formal method, namely B. The focus of the survey was to identify the important features that are necessary for verification tools to become usable to users. The survey assessed the tools' usability based on the Cognitive Dimensions of Notations (CD) framework and several criteria suggested by the International Organization for Standardization (ISO). Sixty-three participants responded to the survey. The data was analyzed by using the grounded theory. **Results:** The analysis enabled the identification of abstract concepts and properties that formed a design guideline for usable verification tools. The guideline includes there main aspects; Interface, Utilities and Resources Management. **Conclusion:** The guideline acts as a roadmap for tool designers to design verification tools that promote the use of formal notations.

Key words: Usability requirement, formal verification tools, empirical assessment

INTRODUCTION

Conceptual models specify the characteristics of the existing and future systems. They are mainly produced through the use of a designated modeling notation. Some examples of the existing notations include graphical notations such as Entity-Relationship Diagram (ERD) (Chen, 1976) and Unified Modeling Language (UML) (Object Management Group, 2010) and formal notations such as Z (Spivey, 1992) and B (Abrial *et al*., 1996). In addition, there are also notations that integrate both graphical and formal such as UML and Z (Martin, 2003) and UML and B (Snook and Butler, 2006).

While graphical notations use visual objects, formal notations use mathematical symbols and interpretation to describe a system. The formality imposed by formal notations enables a model to be verified systematically by tools, which are designed specifically to increase model precision and consistency. In the case of B method for instance, this is achieved by using B verification tools such as ProB (Leuschel and Butler, 2003), Atelier-B (ClearSy, 2003) and B-Toolkit (B-Core(UK) Ltd, 1999). The verification process begins when a set of system requirements are specified using the B notation. Later, the specification is feed into the verification tools for syntax and semantic checking.

Whilst having the ability to increase a model's precision and consistency, formal notations however are regarded as being difficult to comprehend (Carew *et al*., 2005). Indirectly, the verification tools are expected to overcome this barrier. In a sense, they are assumed to be usable and useful. This study 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 useful and usable. The survey employed the Cognitive Dimensions of Notations (CD) (Green, 1989; Green and Petre, 1996) framework with several usability criteria suggested by the International Organization for Standardization (ISO) (ISO/IEC 9126-1, 2001) as its instrument. The instrument was used to guide the discovery of features rather than as a means of assessment of the tools.

Corresponding Author: Rozilawati Razali, School of Computer Science, Faculty of Information Science and Technology, University Kebangsaan Malaysia, 43600 UKM Bangi, Selangor, Malaysia

Background: Formal methods are defined as methods that impose the use of mathematically based approaches to software development. They are seen as a fault avoidance technique that aims to reduce the introduction of errors into a system. The methods are employed at the early stages of system development, particularly from the specification stage.

There are many instances of formal methods, which one of them is the B method (Abrial *et al*., 1996). The B method is a collection of mathematically based techniques for the specification, design and implementation of software components. It provides techniques that ensure the consistency of a specification and guarantee the implementation with respect to that specification. There are two main verification activities involved in the B method; 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 verification activities involving the B method. For instance, BToolkit by B-Core(UK) Ltd (1999). Figure 1 depicts an example of B-Toolkit interface. Such 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. Some complex proof obligations however 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 behaviors.

Besides the industrial tools, there are also tools developed within the research community. ProB (Leuschel and Butler, 2003) for instance, supports the automated Consistency and Refinement Checking via Model Checking (Clarke *et al*., 1999). Unlike other B tools, ProB comprises a model checker that explores exhaustively the finite behavior 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. The animations allow the simulated behavior of a model to be observed. In particular, 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.

Fig. 1: A screenshot of B-Toolkit (B-Core(UK) Ltd, 1999)

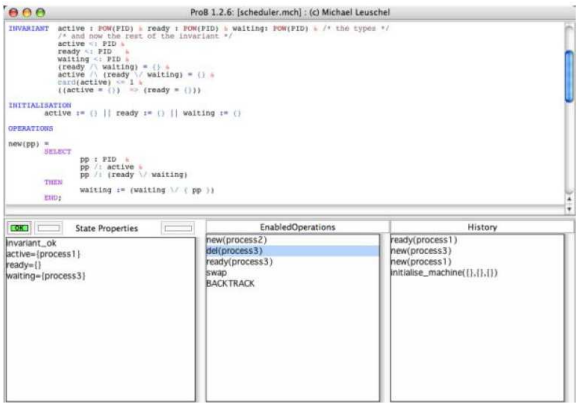

Fig. 2: A screenshot of ProB 1.2.6

Figure 2 shows the interface of ProB 1.2.6, which was used as the object of study in the survey. ProB has recently been upgraded to ProB 1.3.0 (ProB, 2010). Despite this fact however, the findings of the survey are still valid since the analysis was not meant to evaluate the tool per se. Instead, it was intended to capture and generalize important features that must be catered by verification tools, based on the feedback received from the respondents.

MATERIALS AND METHODS

The objective of the survey was to capture some experience of using verification tools that support a formal notation in conceptual modeling. 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 present in verification tools for them to become useful and usable. The survey started the investigation with two instances of verification tools, namely ProB and BToolkit. As a study on two instances could not reveal all features, the findings from the survey are left open for further investigation and discussion in future where they can be validated and expanded.

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?

The survey instrument was based on the Cognitive Dimensions of Notations (CD) usability framework (Green, 1989; Green and Petre, 1996). The framework comprises fourteen dimensions as illustrated in Table 1 below, which acted as the variables in the survey. In addition, several usability criteria of ISO were also included. The CD framework was adopted because it is a tool that aids the usability evaluation of information-based artifacts (Green and Blackwell, 1998), which formal specifications are one of such artifacts. As a usability tool, it concentrates on the processes by considering the perspective of people who deal with the artifact and its environment.

There are many different approaches to dealing with qualitative data employed in the social sciences (Cassell and Symon, 1994; Denzin and Lincoln, 1994; Strauss and Corbin, 1998). The survey adopted one approach, namely the grounded theory (Strauss and Corbin, 1998) because it is systematic and directive. It contains structured procedures to generate theories based on the stated research question. The questions for the survey were constructed by following the proposed CD questionnaire (Blackwell and Green, 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 used an ordinal scale that provided the respondents with seven potential levels of agreement, from -3 (very difficult) to 3 (very easy). 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.

Table 1: Cognitive dimensions (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 presented in a similar syntactic manner
Diffuseness	Complexity or verbosity of the are 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 s simultaneously or view two related components side by side at a time

To illustrate briefly the questions, below are some examples. The first question concerns the visibility 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/learns ability criteria.

How easy is it to view and search the various features in ProB/BToolkit when you are working with your B model?

Very difficult						Very easy
-3	-2	-1	0	1	2	3

Why?

Does ProB/BToolkit 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?

If you verify your B model in ProB/BToolkit, how difficult is it to comprehend what is happening?

Very Difficult						Very easy
-3	-2	-1	0	1	2	3

Why?

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.

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 Master students of Computer Science and Software Engineering courses from two universities in the south of England. Master students constituted one-third of the participation. Non-British students 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. The participants had gone through courses on formal methods at some points of their studies. The participants were in the final semester of their respective courses and thus had reasonable amount of experience and knowledge of software development. Some of the Master students had some industrial experience for at least one year.

The participation was voluntary where the questionnaires were completed anonymously and submitted at the end of the semester. The participants were aware that the survey was intended for research purposes. The survey adhered to the ethical policies and guidance for conducting research involving human participants. In particular, the materials and procedure used in the survey had been reviewed and approved by the institution's Ethics Committee.

RESULTS

Due to the extensiveness and confidentiality of the data, they are not presented in this study (Readers may obtain the raw data by contacting the corresponding author at rozila@ftsm.ukm.my). This study however discusses the findings of the analysis. 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. 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.

A set of feature properties have been identified from the data. 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, Utility and Resources Management.

Table 2: Properties and dimensions of "Interface"

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)
Panes	Width of panes can be resized
	Panes can be closed/collapsed/minimized and reopened/expanded/maximized
	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)
	Use conventional buttons with standard meanings (i.e., <OK> to confirm and <CANCEL> to defer)
	Not all dialogues should be closed to proceed (e.g., online help windows can be displayed and remain while model editing)

Utility is the main functionality of verification tools. To perform as intended, Utility requires Interface and Resources Management. The properties of Utility are therefore interrelated with the properties of the other two categories. The following paragraphs list the categories and properties. The corresponding interrelated properties are stated in the parentheses in the table of Category 2 (C2): Utility.

Category 1 (C1): Interface: This category refers to the structure and organization of screen layout and utilities. Table 2 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 modeling 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 such as the use of toolbar and short-cut keys. Moreover, the utilities must represent closely the principles of the underlying formal methods so that users can smoothly apply the methods. The utilities should be controlled by the way they should be used. As formal modeling is mainly rigid, users should be offered with supporting utilities that can be set as needed. This is to ease the 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. 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 modeling due to the complexity of the tasks.

Category 2 (C2): Utility: This category refers to the utilities required for formal modeling. Table 3 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.

Table 3: Properties and dimensions of "Utility"

Property	Dimension
Editing and formatting	Text can be formatted (i.e., size, color), edited (i.e., cut, paste, undo, redo) and searched (i.e., find and replace, go to) (C1: Menu)
	Model/file can be organized and manipulated (i.e., save as different file, print) (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) 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 color coding to demonstrate animated elements (C1: Menu; C3: Interoperability)
	Animated elements can be viewed easily (i.e., zooming, side-by-side) 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 color 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)

Being able to check the accuracy and consistency of a model is the main advantage of formal modeling. 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 behavior under the stated conditions and rules. The facility may be available in several different mediums and can be done automatically and semiautomatic ally. Automatic animation is only feasible for accurate and consistent models. Therefore, semiautomatic animation is useful for users to identify specific points where rules violation and unintended behaviors 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 color 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 someaspectsofthe task cannot be performed automatically.

Table 4: Properties and dimensions of "Resources Management"

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

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 visualization. Several different approaches may be available for users to verify the model. Visual indicators such as colors 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 facilitate 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.

Category 3 (C3): Resources management: This category refers to the management of entities that are related to the execution of utilities. Table 4 lists the necessary properties and dimensions.

Users should be given several options of running the tools. The tools should cater several different Platforms 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. This is called refinement. 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 visualization 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 to 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.

DISCUSSION

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. To improve the accuracy of the guideline, further investigation and discussion are needed so that it can be confirmed and refined.

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.

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

Instrument: The survey aimed to discover as many features as possible that can ensure the usability of verification tools. It employed the CD framework and several usability criteria of ISO as its instruments. The instruments used 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.

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.

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 courses and had reasonable amount of experience and knowledge of software development. Moreover, 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.

Toy problem: The coursework that required the respondents to use the tools was not large. However, the coursework was believed to be sufficient for the respondents to experience the tools and verification tasks.

Dependent variables: The dependent variables of survey were the fourteen dimensions of CD and four usability criteria of ISO. They survey might have used other variables. But, 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.

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 modeling task could have reduced this threat, as the respondents still remembered of what he or she found during the task.

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.

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 may not represent all possible verification tools. The findings of this survey should be thus confirmed and refined in future studies.

CONCLUSION

This study 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 enabled the identification of abstract concepts and properties of usable verification tools. The concepts and properties formed 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 organizes 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. 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. 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.

ACKNOWLEDGEMENT

This study was funded by the Malaysian Government and University Kebangsaan Malaysia. The author thanks all the participants who responded to the survey.

REFERENCES

Abrial, J.R., A. Hoare and P. Chapron, 1996. The B-Book: Assigning Programs to Meanings. 1st Edn., Cambridge University Press, Cambridge, ISBN: 10: 0521496195, pp: 813.

B-Core(UK) Ltd., 1999. B-Toolkit. B-Core(UK) Limited. http://www.b-core.com/btoolkit.html

Blackwell, A.F. and T.R.G. Green, 2000. A cognitive dimensions questionnaire optimized for users. Proceedings of the 12th Annual Meeting of the Psychology of Programming Interest Group, Apr. 2000, PPIG Press, Cozenza Italy, pp: 137-152. http://www.ppig.org/papers/12th-blackwell.pdf

Carew, D., C. Exton and J. Buckley, 2005. An empirical investigation of the comprehensibility of requirements specifications. Proceedings of the International Symposium on Empirical Software Engineering, Nov. 18-18, IEEE Computer Society, Noosa Heads, Qld., pp: 256-265. DOI: 10.1109/ISESE.2005.1541834

Cassell, C. and G. Symon, 1994. Qualitative Methods in Organizational Research. 1st Edn., Sage Publication, Thousand Oaks, CA., ISBN: 10: 0803987706, pp: 266.

Chen, P.P., 1976. The entity-relationship model: Toward a unified view of data. ACM Trans. Database Syst., 1: 9-37. DOI: 10.1145/320434.320440

Clarke, E.M., O. Grumberg and D.A. Peled, 1999. Model Checking. 1st Edn., MIT Press, USA., ISBN: 10: 0262032708, pp: 314.

ClearSy, 2003. Atelier B user manual V3.6. ClearSy System Engineering. http://www.clearsy.com

Denzin, N. and Y. Lincoln, 1994. Handbook of Qualitative Research. 2nd Edn., Thousand Oaks, Sage Publication Ltd., CA., ISBN: 10: 0761915125, pp: 1143.

Green, T.R.G. and A.F. Blackwell, 1998. Design for usability using cognitive dimensions. Proceeding of the Tutorial Session at the British Computer Society Conference on Human Computer Interaction, (HCI'98), Sheffield, UK., pp: 1-75. http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/CDtutorial.pdf

Green, T.R.G. and M. Petre, 1996. Usability analysis of visual programming environments: A cognitive dimensions framework. J. Vis. Lang. Comput., 7: 131-174. DOI: 10.1006/jvlc.1996.0009

Green, T.R.G., 1989. Cognitive Dimensions of Notations. In: People and Computers V, Sutcliffe, A. and L. Macaulay (Eds.). Cambridge University Press, Cambridge, pp: 443-460.

ISO/IEC 9126-1, 2001. Software engineering, product quality-part I: Quality model (Standard No. 9126-1). International Organization for Standardization. http://www.iso.org/iso/catalogue_detail.htm?csnumber=22749

Leuschel, M. and M. Butler, 2003. ProB: A model checker for B. Lecture Notes Comput. Sci., 2805: 855-874. DOI: 10.1007/978-3-540-45236-2_46

Martin, S., 2003. The best of both worlds integrating UML with Z for software specifications. J. Comput. Control Eng., 14: 8-11. DOI: 10.1049/cce:20030201

Object Management Group, 2010. Introduction to OMG's Unified Modeling Language (UML). Object Management Group. http://www.omg.org/gettingstarted/what_is_uml.htm

Snook, C. and M. Butler, 2006. UML-B: Formal modeling and design aided by UML. ACM Trans. Software Eng. Methodol., 15: 92-122. DOI: 10.1145/1125808.1125811

Spivey, J.M., 1992. The Z Notation: A Reference Manual. 2nd Edn., Prentice-Hall, Englewood Cliffs, ISBN: 13: 9780139785290, pp: 150.

Strauss, A.L. and J. Corbin, 1998. Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. 2nd Edn., Sage Publication, Thousand Oaks, California, ISBN: 0803959397, pp: 366.

ProB, 2010. The ProB animator and model checker. ProB. http://www.stups.uniduesseldorf.de/ProB/