An Empirical Investigation to Understand the Difficulties and Challenges of Software Modellers When Using Modelling Tools

Parsa Pourali
Department of Electrical and Computer Engineering
University of Waterloo, Canada
ppourali@uwaterloo.ca

Joanne M. Atlee
David R. Cheriton School of Computer Science
University of Waterloo, Canada
jmatlee@uwaterloo.ca

ABSTRACT

Software modelling is a challenging and error-prone task. Existing Model-Driven Engineering (MDE) tools provide modellers with little aid, partly because tool providers have not investigated users' difficulties through empirical investigations such as field studies. This paper presents the results of a two-phase user study to identify the most prominent difficulties that users might face when developing UML Class and State-Machine diagrams using UML modelling tools. In the first phase, we identified the preliminary modelling challenges by analysing 30 Class and State-Machine models that were previously developed by students as a course assignment. The result of the first phase helped us design the second phase of our user study where we empirically investigated different aspects of using modelling tools: the tools' effectiveness, users' efficiency, users' satisfaction, the gap between users' expectation and experience, and users' cognitive difficulties. Our results suggest that users' greatest difficulties are in (1) remembering contextual information and (2) identifying and fixing errors and inconsistencies.

CCS CONCEPTS

• **General and reference → Empirical studies**; • **Software and its engineering → Software usability**; **Software notations and tools**; • **Human-centered computing** → *Empirical studies in HCI*;

KEYWORDS

Human-Centric Software Development, Empirical Study, UML, Model-Easing, Modelling Tools.

ACM Reference Format:
Parsa Pourali and Joanne M. Atlee. 2018. An Empirical Investigation to Understand the Difficulties and Challenges of Software Modellers When Using Modelling Tools. In *ACM/IEEE 21th International Conference on Model Driven Engineering Languages and Systems (MODELS '18), October 14–19, 2018, Copenhagen, Denmark,* Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3239372.3239400

1 INTRODUCTION

Model-Driven Engineering (MDE) addresses software complexity by raising the level of *abstraction* in software artifacts, and facilitating the *automation* of code generation and software verification [19][32]. However, modellers often find it cognitively difficult to create, edit, and debug models, and they expend a lot of effort on these tasks [14][38][43].

Researchers investigate various tools and methods to reduce the effort of editing and debugging models [13][32][37], but the tools are not adopted because they do not meet the users' needs. The reasons are, in part, that tool designers:

(1) have not identified and understood the difficulties and challenges of users (e.g., through empirical observations);
(2) have not taken into account human-cognition factors that can explain users' difficulties and challenges; and
(3) have conducted few empirical evaluations of the effectiveness of their tools in supporting human users.

We performed a formative user study to learn about modellers' most-severe challenges when using modelling tools and to understand some of the most-critical obstacles to tools adoption. Specifically, we focused on identifying the cognitive challenges that modellers face when designing structural and behavioural models of software systems, as exemplified by the UML Class and State-Machine diagrams. The goal of this work is to help tool researchers and vendors to know where to focus their future tool-building efforts.

We conducted a two-phase user study. In the pre-study phase, we analyzed 30 models (i.e., Class diagrams and State-Machine diagrams) that had previously been developed as solutions to a course assignment. We reviewed the assignments and looked for modelling errors made by the students and looked for evidence of challenges that they faced when doing the assignment. The results obtained from the pre-study phase informed our design of a user study, which investigated modellers' usage of modelling tools, including the tools' effectiveness, users' efficiency, users' satisfaction, the gap between users' expectation and experience, and users' cognitive difficulties. We recruited 18 subjects and ensured that they have sufficient knowledge about the Unified Modelling Language (UML) and have experience of using at least one modelling tool. The subjects were asked to perform seven modelling tasks consisting of developing partial State-Machines of a parking lot system. For each subject, we recorded various User eXperience (UX) metrics such as the subjects' performance and verbal expressions.

The results showed a substantial gap between users' expectations of tools' abilities to alleviate the challenges and the users' actual experience of using the tools. Also, the results revealed modellers' prevalent challenges when using modelling tools, among which 1) remembering contextual information and 2) identifying and fixing errors and inconsistencies are the most-critical and are most in need of consideration from tool vendors.

The rest of this paper is organized as follows. Section 2 presents the related works on empirical studies to evaluate MDE tools. In Section 3 we describe the context of our experimental research. Section 4 explains the design of our user study. Section 5 describes the execution procedure and practical considerations of it. We present the results of our study in Section 6, and discuss some important issues in Section 7. In Sections 8 and 9, we discuss the threats to validity of our user study and conclude our work, respectively.

2 BACKGROUND AND RELATED WORK

While the SE community has recently made efforts to narrow the gap between SE and HCI [1] (e.g., by addressing the challenges of creating usable programming IDEs [20]), the MDE community has focused more on understanding the obstacles to MDE's adoption caused by factors other than tools (e.g., social, organizational, cost, etc.)[17][30][42][47]. For example, Hutchinson et al. [17] use questionnaires and interview methods to survey many practitioners to acquire knowledge on how technical, social and organizational factors impact the adoption of MDE in industry. Similarly, Mohagheghi et al. [30] discuss the challenges (e.g., risks and costs) that companies may undergo to adopt MDE. Also, Whittle et al. [47] put tool-related issues in a broader context of social and organizational factors and present its impacts on the MDE adoption. These works provide insights into challenges that affect different levels of the MDE process, but they do not target the difficulties of using modelling tools as part of the process.

As per challenges of modelling, few works aim at understanding the challenges of model comprehension. For instance, Kuzniarz et al. [23] and Zayana et al. [48] investigate the effectiveness of using stereotypes and examples in improving the comprehension of UML models, respectively. Kutar et al. [22] test whether users' understanding of information in Sequence diagrams differ from their understanding of Collaboration diagrams. These works provide information on users' difficulties in comprehending models, rather than learning about challenges with modelling tools.

Some other works propose novel tooling techniques to help users in developing models. As an instance, Alsuraihi and Rigas [5] empirically evaluate how auditory techniques (e.g., voice input) can ease the task of editing modelling. Furthermore, some works such as ArgoUML [43], Yakindu [31], MagicDraw [18], Pati et al. [38] and Dyck et al. [9] introduce heuristic tooling techniques such as model assist, recommenders, graphical decorators and dialogue boxes to reduce the cognitive loads of editing models. With respect to easing the task of model-debugging, some researchers have proposed tooling techniques for automatic detection and repair of inconsistencies among related models. For example, Xlinkit [33] is a tool that manages inconsistencies among different documents that are developed in XML. Also, Egyed (UML/Analyzer) [10] and Ramesh et al. (XRTSDIC) [40] propose instant consistency checkers that detect model inconsistencies immediately by observing model changes and checking the changes against a set of consistency rules. Snoeck et al. (MERODE) [46] provides a formal definition of UML diagrams and applies a few simple rules to the check a model's completeness. These works propose useful features to reduce the effort of developing models, but never assess the effectiveness of their solutions on users. As such, it is unknown whether their features have difficult-to-use user interfaces (UIs) for MDE tools, which is a barrier to MDE adoption [1][32].

The empirical evaluation by Hadar and Zamansky [12] and Huang et al. [16] are the works most related to ours with respect to understanding modellers' challenges. Hadar and Zamansky [12] interview several subjects to understand their perception of the cognitive challenges in resolving inconsistencies and identify the determining cognitive factors. However, they do not analyze the subjects when using any tools. Huang et al. [16] take a systematic approach to investigate users' interactions and identify their challenges with MetaSketch tool [34]. The goal of their work is very similar to ours, but we take a more comprehensive approach and investigate on a broader subset of modelling tasks and tools.

Although all of the aforementioned works contribute to the MDE ecosystem, they do not empirically analyse modellers and their tasks when using tools to develop models. Much of the prior work on identifying issues with modelling tools are based on surveys or interviews rather than user studies (e.g., [47]). Our work aims at improving the MDE ecosystem by identifying prominent modelling challenges and proposing novel tool advances.

3 EXPERIMENTAL CONTEXT

Models can be developed in various modelling languages. Some companies use their own Domain-Specific Language (DSL) whereas others prefer to use more general-purpose languages such as the Unified Modelling Language (UML) or Systems Modelling Language (SysML). The majority of models that are being developed in industrial practice are based on the UML or UML-like notations as the UML has become the de-facto standard for modelling software systems [24] and is actively taught and used by academics.

The UML consists of several diagrams that can be partitioned into two types: *static* and *dynamic* [35]. It would be too time-consuming to cover all diagram types in a single user study, thus, we confined the scope of our study to one important static diagram and one important dynamic diagram [11], in particular Class diagrams and State-Machine diagrams. Hereafter, the term *modelling* refers specifically to Class diagram and State-Machine diagram modelling.

We investigated the following research questions:

- *RQ.1: How effective are tools in communicating with users to improve the experience of performing modelling tasks and the correctness of models?*
- *RQ.2: How efficient are modellers when using modelling tools?*
- *RQ.3: How well do modelling tools meet users' expectations?*
- *RQ.4: Overall, how satisfied are users with modelling tools?*
- *RQ.5: Which challenges are the most severe experienced by modellers employing modelling tools?*

The research questions were investigated by means of a two-phase user study. In the first phase (referred to as the *pre-study*), we conducted a lightweight analysis of a set of existing models developed as part of a course assignment, and we looked for common modelling errors made by the modellers as well as evidence of challenges that the modellers faced. Then in the second phase,

Table 1: Summary of the Results of the Pre-Study Analysis

Ctgry. ID	Description	No. of Errors	No. of Subjects
Ctgry. 1	Incorrect use of the structure of UML models (e.g., a State-Machine without an initial pseudo-state).	7	4
Ctgry. 2	Referring to an undefined variable or entity. This includes misspelling the name of an existing variable or entity, or writing incorrect paths in navigation expressions.	248	24
Ctgry. 3	Wrong or inconsistent use of UML notation and syntax.	27	8
Ctgry. 4	Type mismatch between the left-hand-side (LHS) and right-hand-side (RHS) of an assignment or condition.	42	9

we used the results of the pre-study to limit the scope of the user study to model-editing and model-debugging tasks that were most likely to be problematic. For example, we asked nothing in the user study about the structure of a State-Machine or about setting the names of the states because the results of our pre-study showed that most of the subjects could successfully manage such tasks.

3.1 Pre-Study Phase

In the pre-study phase, we examined 30 models that had been submitted as solutions to a modelling assignment in an upper-year undergraduate course on *Software Requirements: Specification and Analysis* at the University of Waterloo. The students were asked to design a State-Machine diagram for a parking-lot system based on a given domain description and domain model. Student could develop their models using a modelling tool or a drawing tool. There were no time constraints except the assignment deadline. The course lectures and readings covered the necessary knowledge on the relevant UML modelling, especially Class and State-Machine diagrams. Moreover, students could seek help from the course instructor if they faced any problems understanding the course materials, and they could consult UML resources and documentation.

We assessed the models based on the marking scheme set by the course instructor. The marking scheme helped us evaluate the models from two aspects:

(1) *Information Content* – detecting inconsistencies between the given textual description of the system and the submitted model. We used this as a guideline to estimate how much of the modellers’ difficulties actually laid in expressing the domain/system description in the modelling notation.
(2) *Model Quality* – detecting errors related to the well-formedness, correctness, and consistency of the models. To be more rigorous, we also assessed the models with respect to a taxonomy of error types proposed by Lange et al. [24].

3.1.1 Pre-Study Results. Evaluation of the models’ Information Content determined that only four students submitted models that were incomplete with respect to the provided domain description. This suggests that most students were able to represent the problem description as a basic UML State-Machine that informally captured all of the described behaviour.

However, the evaluation of Model Quality suggests that students had difficulty creating correct and consistent models at the expected level of detail and precision. We grouped Model-Quality errors into different categories, listed in Table 1. The Table presents the number of subjects that committed errors of each error type, and how many instances of each error type were made in the 30 models. In some cases, such as *Ctgry. 2*, a large number of mistakes were made – often the same mistake was made multiple times. For example, a student would refer to an undefined element over and over without noticing that there was no such element in the Class diagram.

Table 2: Modelling Challenges Identified in the Pre-Study

Name	Description
Order	Performing a sequence of actions in the right order.
Context	Remembering contextual information (e.g., consulting related diagrams to remember names and associations).
Navigation	Writing navigation expressions (navigating correctly from one model element to a related model element).
Syntax	Remembering the keywords and syntax of the language.
Type-Matching	Matching the types of the LHS expression and RHS in an assignment (=) or a condition (==).
Debugging	Locating, understanding, and resolving errors. This includes switching back and forth among multiple diagrams to fix an inconsistency.

The outcome of the Pre-Study was a preliminary list of modelling challenges referred to as *pre-study challenges* (see Table 2). Because students made few Category1 errors (structural errors), we did not consider these to be a significant modelling challenge. In contrast, a quarter of the students or more made errors in Categories 2-4, thus we deemed these to be significant enough to include a pre-study challenge. In addition to these challenges, Reason [41] notes that some errors can be in the form of *slips or lapses,* which result from failing to execute all steps in a sequence of model edits (e.g., creation of a new state machine should be followed by creation of an initial state, a pseudo-state that refers to the initial state, at least one transition, etc.). Accordingly, we include *Order* in our list of challenges. We added *Debugging* as a significant challenge because we observed that some errors could be simply avoided if a student invoked the tool’s debugging feature. Subsequently, we scoped the tasks in our main study to focus on these suspected challenges.

4 EXPERIMENTAL DESIGN

In order to analyze and learn about the challenges that modellers face, we conducted a *formative* empirical study rather than a *summative* one. We explain the design of our study in this section.

4.1 Recruitment Procedure

We emailed a recruitment letter[1] to invite interested subjects to fill out a questionnaire using SurveyMonkey. The questionnaire asked subjects about their demographic, professional and academic backgrounds. The letter went to all the students in the programs of Software Engineering, Computer Science, and Electrical and Computer Engineering; we considered these students a good fit

[1]All of the materials regarding the user study can be found at [39].

Table 3: Subjects' Demographics and Backgrounds

Category	Sub-Category	Count
Occupation	Graduate Student	14
	Post-Doc Researcher	2
	Software Engineer (Industry)	2
UML Familiarity	Fairly Familiar (Novice)	1
	Familiar	6
	Very Familiar	8
	Strongly Familiar (Experienced)	3
Experience with Tools	One to six months	6
	Seven to 12 months	4
	One year to two years	2
	More than two years	6

Table 4: Number of subjects per tool.

StarUML	ArgoUML	Visual Paradigm	MagicDraw	UmLet	Umple	Papyrus	Astah
3	3	3	3	1	2	2	1

for our study because they take modelling courses as part of their program. The letter also went to graduate students with experience in software engineering research including modelling. Additionally, we distributed flyers around campus to reach possible non-student subjects such as post-docs, alumni, or even subjects from industry.

The Screening Procedure: To help ensure that our subjects were representative of the larger population of UML modellers, we designed a screening questionnaire to collect information about their knowledge of the UML. This questionnaire included multiple-choice questions that asked subjects to rank on a Likert scale [28] their familiarity with UML Class and State-Machine diagrams. In addition, it included 10 UML-specific questions selected from online sample practice tests such as the Sun Certified Java Associate exams. Only those respondents who expressed some familiarity with the UML and who passed all 10 test questions were considered eligible for the study and were invited to schedule a study session. The other respondents were contacted and informed of their ineligibility.

Population: We aimed for a study with 20 subjects [4],[21],[44]. However, after almost six months of recruiting, we enlisted 18 subjects. An overview of the subjects' occupation, UML familiarity and tool experience is provided in Table 3.

4.2 The Application Domain

To minimize the effects of domain knowledge on the subjects' performance on tasks, we designed the study around a fairly simple application domain: a gated parking-lot system. Moreover, to familiarize the subjects with the application domain, we asked the subjects to study a textual description of the domain as well as the Class diagram of the system before starting the study's tasks.

4.3 Treatment Allocation

To guard against the threat to validity that poor performance could be due to unfamiliarity with a specific modelling tool, we allowed the subjects to use the modelling tool of their choice. Table 4 shows a summary of the distribution of the tools amongst subjects. It is notable that most of the tools that were chosen by our subjects are among the list of most-heavily used tools reported in a recent survey by Anger and Lethbridge [2].

4.4 Tasks

Each subject was given a textual description and a partial Class diagram of a parking lot system, and was asked to edit and debug variants of a State-Machine diagram. The names used in all the diagrams (e.g., class attributes' names) were chosen to ease comprehension of the model. Also, the researcher was present during the study to answer the subject's questions about the domain description, or clarify the tasks, if needed.

The experiment comprised seven tasks. The first four tasks were designed to gauge the effort of editing models (e.g., developing State-Machines and editing transition expressions), whereas the last three tasks were designed to understand the challenges of users when debugging models (i.e., finding and fixing errors and inconsistencies in the models). We designed simple tasks mainly for two reasons: 1) to increase the size of the pool of potential eligible subjects, and 2) to ascertain whether challenges exist even for such simple tasks, let alone for complicated tasks.

Model-Editing Tasks: In each of the first four tasks (i.e., *Task1, Task2, Task3, and Task4*), the subjects were given a structured textual description of a transition and were asked to use the modelling tool to set the *triggering event*, *guard*, and *action* of the transition. Below is an example of a model-editing task.

Task1: Please develop the transition that is labelled as T1 in the diagram (based on the following description).

- *Event: No triggering event is required for this transition.*
- *Guard: If the gate id is B.*
- *Action: The Gate will go to the closed state; that is, the gate position should be set to down.*

Model-Debugging Tasks: For each of the three debugging tasks, we introduced a few inconsistencies in the diagrams and asked the subjects to locate and fix them. They could either examine the diagrams manually or use the tool's diagnostic features. Specifically, *Task5* asks the subjects to rename elements in the Class diagram, and then locate any inconsistencies in the model that were introduced by that action. In *Task6* and *Task7*, the subjects were asked to locate inconsistency errors that were embedded in the model, such as identifying model elements that were used but not defined, and detecting incorrect navigation expressions. Following is a sample model-debugging task (i.e., *Task5*).

Task5: Assume that you are supposed to change the name of the gates from A and D to GA and GD respectively in the Class diagram. Please implement the change in the model and report any inconsistencies you found that are caused by this change.

4.5 Data-Collection Techniques and Design

We evaluated the subjects’ performance along three different dimensions, as prescribed by Tullis and Albert [4], namely performance, self-reported, and behavioural metrics. This section precisely defines the variables and metrics measured in the study.

4.5.1 Performance Metrics. We measured three performance metrics:

- **Task Completeness:** We measured the degree to which a subject was effective in completing a task. We defined four levels of success:

(1) *Complete(1.0)*: A subject completed a task without any assistance. Note that, in model-editing tasks, a score of Complete does not mean that the task was error free. A model-editing task is deemed Complete if no errors of omission were performed. Errors of commission are possible. A model-debugging task is deemed Complete only if all errors are found and fixed.
(2) *Partially Complete (0.5)*: A subject asks for help during the task, such as asking about the language syntax, or asking for clarification about a model element.
(3) *Incomplete (0.0)*: A subject was unable to complete a task. For model-editing tasks, a score of Incomplete means that a subject omitted some aspects of a task’s requirement, whereas in model-debugging tasks it means a subject could not locate all of the embedded errors in the model.
(4) *Generally Complete*: Refers to the summation of the above three scores for a task (i.e., +1.0 for each subject with a Complete score and +0.5 for each subject with a Partially Complete score).

- **Errors:** We counted the number of errors committed by each subject per task and classified errors as being either *well-formedness* or *consistency* errors [24]. We did not count *completeness* errors as our tasks were not designed to analyze these types of errors. It is important to note that the number of errors is different from the degree of task completeness as a task can be completely done but not be error-free.
- **Efficiency:** We measured efficiency using two metrics:

(1) *Time-on-Task* refers to the time that it takes a user to perform a particular task using a product, and is comparable to Hill’s definition of modelling effort [15].
(2) *Lostness* measures how *"lost"* a subject is when performing a task. To assess lostness, the following three factors are measured: 1) R: the minimum number of diagrams or dialogues that must be visited to accomplish the task, 2) S: the total number of diagrams or dialogue-boxes visited while performing the task (counting revisits), and 3) N: the number of different (unique) diagrams or dialogue-boxes the subject visited while performing the task. Lostness, L, is then calculated using the following formula [4][45].

$$L = \sqrt{(\frac{N}{S} - 1)^2 + (\frac{R}{N} - 1)^2} \tag{1}$$

Lostness scores range from *Zero* to *One*. The higher the score, the more trouble the user had finding what they want. Smith [45] found that users with a lostness score of less than 0.4 have no substantial difficulty to fulfill a task, whereas users with a lostness score of greater than 0.5 are definitely lost. One can also estimate subjective lostness by comparing to the optimal score (e.g., to the smallest value of lostness among all the subjects) [4].

4.5.2 Self-Reported Metrics. Perhaps the most traditional means of assessing the usability of a tool is asking users to tell us about their expectation and experience with the tool [3][4]. We designed our experiment to collect self-reported data from users to gauge their satisfaction.

Expectation versus Experience ratings: We asked the subjects to rate two things:

(1) their *Expectation Rating* of how easy or difficult they expected a task to be (based on their understanding of the task and the tool) before starting the task, and
(2) their *Experience Rating* of how easy or difficult each task actually was.

We used the same 7-point rating scale (1=Very easy to 7=Very difficult) for both ratings. For each task, we then calculated an average *Expectation Rating* and an average *Experience Rating* of all the subjects. The difference in the two average ratings indicates a degree to which the tools are effective in satisfying the users’ expectations and needs: a difference score of *zero* indicates high effectiveness in satisfying the users’ expectations, and a difference score of 6 suggests an imbalance between the users’ expectations and what the tools provide.

The expectation and experience ratings were collected in two different stages of the study: 1) before and after performing a task, and 2) at the beginning and at the end of the study.

- *Pre-Session Expectation Rating:* Using the Pre-Session Expectation Rating, we collected data about the subjects’ overall expectation of the tools’ proficiency with respect to the pre-study challenges before starting the session (Fig. 2).
- *Post-Session Experience Rating:* Once the session was completed, we gave the same set of questions from the Pre-Session Expectation Rating to the subjects, asking their opinions about how well the tool provided features that aided them and how well the tool met their expectations to overcome the pre-study challenges. We can use the averages of the Pre- and Post-Session ratings to get insight into opportunities for improving the tools.
- *Pre-Task Expectation Rating:* Before starting each task, we asked subjects to rate how easy or difficult they thought each task should be based on their expectations of the tool and understanding of the task. The rating was a 7-point Likert-scale from 1=Very easy to 7=Very difficult.
- *Post-Task Experience Rating:* Once each task was finished, we asked users to rate how easy or difficult the task was based on their actual experience of using the tool.

Usability Questionnaire: In addition to the expectation and experience ratings, we gave subjects a usability questionnaire to collect information on the subjects’ satisfaction with the usability of their respective tools. Different usability questionnaires could be employed for this assessment, such as System Usability Scale (SUS) [7], Computer System Usability Questionnaire (CSUQ) [27], Questionnaire for User Interface Satisfaction (QUIS) [8], and, Usefulness, Satisfaction, and Ease of Use Questionnaire (USE) [29]. Among these, we decided to use CSUQ [26][27] because the questions listed in the CSUQ were more in line with the goals of our study. It needed almost no adaptation, whereas the other questionnaire types would have needed much more adaptation to their questions. The adaptation that we made in the CSUQ questions were to replace the term *"system"* with the term *"tool"* in the questions. Our CSUQ consisted of 19 statements to which the user rated agreement on a 7-point scale of "Strongly Disagree" to "Strongly Agree", plus N/A.

4.5.3 *Behavioural Metrics.*

The think-aloud protocol [6] allowed us to collect information about usability issues from the subjects' verbal statements such as:

- Verbal expressions of confusion, frustration, dissatisfaction, pleasure, or surprise.
- Verbal expressions of confidence or indecision about a particular action that might be right or wrong.
- Not saying or doing something that should have been said or done.

To get the most out of the subjects, we prompted the subject if he/she did not express his/her thoughts loudly. The prompts were based on the situation, but some examples are: *What are you thinking now? What are you trying to do? Why did you do that?* To help the accuracy of our data (i.e. time on tasks), we tried to avoid too much prompting and tried to have minimal discussion when the subject was performing a task. After each session, we analysed the audio recording of the subject's session to identify the most valuable verbal expressions, expanding the list of the significant verbal expressions from all subjects as we processed subjects' sessions. We coded and classified verbal expressions into different categories. By counting the number of times that the subjects made verbal statements within each category, we could obtain useful results about the prominent challenges that the subjects faced.

5 EXECUTION AND PRACTICAL CONSIDERATIONS

The study was conducted in 18 separate sessions, one for each subject. The subjects performed their tasks using a PC machine in the researcher's office. The duration of each session ranged from one hour to nearly two hours with an average of about 80 minutes. Also, to automate the process of data collection, we developed a tool that automatically records the time on each task (as a measure of effort). The time on each task starts from the time that the subject begins the task (including reading the description of the task) and ends when the subject acknowledges that he/she is done with the task (i.e., presses the *done* button in the tool). Also, the tool stores other data such as responses to all questionnaires.

The study consisted of several segments: Preparing the Subject, Collecting the Pre-Session and Pre-Task Expectation Ratings, Performing the Tasks, Collecting the Post-Session and Post-Task Experience Ratings, and conducting the CSUQ. Fig. 1 illustrates the structure of a session.

Preparing the Subject: After greetings and signing the consent form, each subject was given an introduction to the study by viewing a preparation video. The subject was asked to watch and listen to the video carefully. The video included general information about the procedure and methods of the study (e.g., think-out-loud method). The advantage of using one video for all the subjects was that they all received the same information with respect to preparing for the study. Moreover, in the video we emphasized that: *in the course of the experiment, the subject will not be evaluated in any way.* We made it clear that it is the tool that is under scrutiny and not them. We needed the subject to understand this because it was important to create a relaxing and informal atmosphere to make the session as effective as possible.

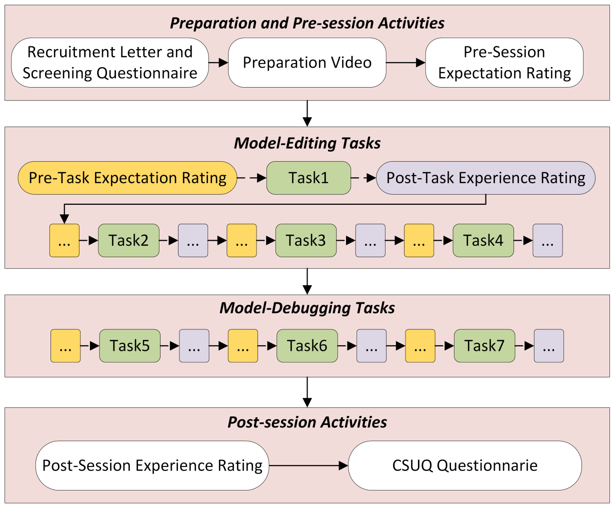
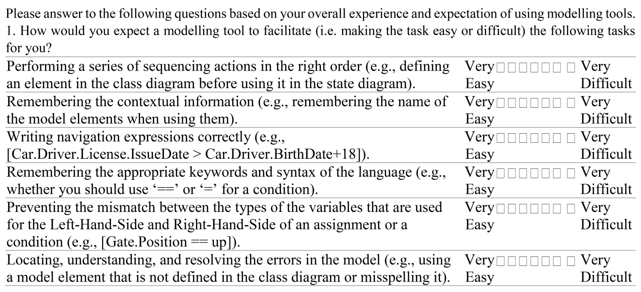

Preparation and Pre-session Activities
Recruitment Letter and Screening Questionnaire
Preparation Video
Pre-Session Expectation Rating
Model-Editing Tasks
Pre-Task Expectation Rating
Task1
Post-Task Experience Rating
...
Task2
Task3
Task4
Model-Debugging Tasks
Task5
Task6
Task7
Post-session Activities
Post-Session Experience Rating
CSUQ Questionnarie

Figure 1: The work-flow of a subject's session.



Please answer to the following questions based on your overall experience and expectation of using modelling tools.
1. How would you expect a modelling tool to facilitate (i.e. making the task easy or difficult) the following tasks for you?

Performing a series of sequencing actions in the right order (e.g., defining an element in the class diagram before using it in the state diagram).	Very Easy	☐☐☐☐☐☐ ☐	Very Difficult
Remembering the contextual information (e.g., remembering the name of the model elements when using them).	Very Easy	☐☐☐☐☐☐ ☐	Very Difficult
Writing navigation expressions correctly (e.g., [Car.Driver.License.IssueDate > Car.Driver.BirthDate+18]).	Very Easy	☐☐☐☐☐☐ ☐	Very Difficult
Remembering the appropriate keywords and syntax of the language (e.g., whether you should use '==' or '=' for a condition).	Very Easy	☐☐☐☐☐☐ ☐	Very Difficult
Preventing the mismatch between the types of the variables that are used for the Left-Hand-Side and Right-Hand-Side of an assignment or a condition (e.g., [Gate.Position == up]).	Very Easy	☐☐☐☐☐☐ ☐	Very Difficult
Locating, understanding, and resolving the errors in the model (e.g., using a model element that is not defined in the class diagram or misspelling it).	Very Easy	☐☐☐☐☐☐ ☐	Very Difficult

Figure 2: Pre-session expectation rating questionnaire.

To familiarize the subject with the application domain, we asked the subject to study a textual description of the domain, the provided Class diagram and its model elements (e.g., classes, operations, attributes) along with a description of all the elements. Finally, we asked the subject to rank their expectations of the tool's ability to ease the *pre-study challenges* that were found in the pre-study phase (see Fig. 2).

Performing the Tasks: We then asked the subject to perform tasks using the modelling tool of his/her choice. Before each task, we asked the subject to read the description of the task and rank how easy or difficult they thought it would be when using the tool. When performing the tasks, subject was asked to express their thoughts out-loud, so that we could understand the subject's cognitive difficulties. For the purpose of later analysis, we screen-captured the subject's work with the tool as well as audio-recorded their voice. Finally, once the subject finished a task, he/she was asked to provide a Post-Task Experience Rating for the task.

There is a trade-off between spending time on a task to keep the quality of the solution high and making progress on all the study's tasks. Imposing any time pressure on the subjects could reduce the quality of the their solutions. Thus, we allowed the subjects to work at their own pace and to announce when they completed a task. However, this could result in the times on tasks becoming unrealistically long and useless for further statistical analysis. To overcome this challenge, we pursued the following strategies:

- If a subject insisted on solving the task but did not show any signs of progress after ten minutes, then the researcher stepped in and asked if the subject needed help (*"Do you need any help or have any question that I can help?"*). In this case, the task success was deemed at best Partially Complete (0.5). If after given hints the subject still could not fulfill the task in the next five minutes, then we asked him/her to move on to the next task and the task was deemed Incomplete (0).
- We did not offer an hourly rate. Our offer was a fixed honorarium of $20 in return for an estimated 90 minutes (maximum) of work for the study. This avoided the threat of subjects playing around with the tool to receive a higher payment.
- To further motivate subjects to finish their tasks in good time, he/she was allowed to immediately leave the experiment once he/she completed the tasks.

Post-Session Activities: At the end of the session, the subject was asked to complete two questionnaires: a Post-Session Experience Ratings and CSUQ[2] [26][27]. The Post-Session Experience Ratings comprised the same set of the questions that were shown in Fig. 2, but this time asked the questions from the perspective of having experience using the tool. The CSUQ collected useful data about the subject's level of satisfaction of using the tools.

In addition, at the end of each session, we asked the subject to provide any additional comments on the tool and their experience of using the tools in an open-ended textual format. This could be positive or negative feedback.

6 RESULTS

This section presents the results of the study with respect to each of the research questions.

6.1 Tools' Effectiveness (RQ.1)

We assessed the effectiveness of the tools in assisting subjects with model-editing and model-debugging tasks by measuring the subjects' success rate and number of errors in their task solutions.

Fig. 3 illustrates the subjects' success rates on the tasks, and shows that a significant number of the subjects were not successful in finishing their tasks unless we provided them with some hints. More importantly, it shows that fewer than 45 percent of the subjects were fully successful in the tasks 6 and 7, which were related to debugging of models. This shows that the tools do not provide enough support to help subjects resolve errors in the models.

[2]A sample CSUQ can be found at http://garyperlman.com/quest/quest.cgi

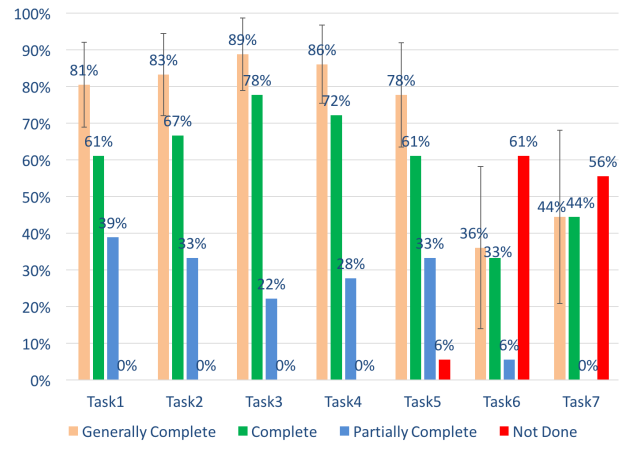


Figure 3: Task completion rate (error bars represent 95% confidence interval).

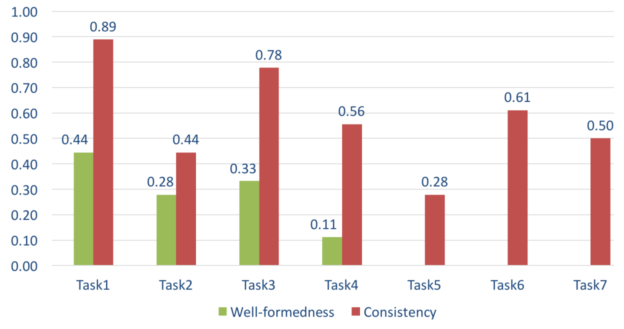


Figure 4: Average error rate made by the subjects per task.

We divided the errors made by the subjects during all the tasks into two classes based on the guidelines given by Lange et al. [24]:

- *Consistency errors* are errors that can be temporarily tolerated but that should be fixed before delivering the model. Consistency errors include: an element is used but not defined, misspelled element names, incorrect navigation paths (e.g., g.blockage instead of g.Sensor.blockage), and type-mismatches between the LHS and RHS of an expression.
- *Well-formedness rules* are UML conventions that can help maximize the model's understandability. Well-formedness errors occurred mostly when a subject used UML syntax incorrectly, or produced an ill-formed expression (e.g., putting extra parentheses or quotations). Well-formedness errors can often be detected through analysis of a single diagram.

Fig. 4 depicts the subjects' error rates for consistency and well-formedness errors per task. For example, in *Task1*, each subject made, in total, 1.33 mistakes. Interestingly, none of the subjects were able to finish all of their tasks without any error. Moreover, more than half of the subjects failed to spot the errors in tasks 6 and 7, where the subjects were asked to debug the models.

This is notable given that the tasks were relatively easy and the model was very small compared to complex industrial models [25]. Based on our observations, the main reason for such high error rates was that the subjects relied too much on the tools, and

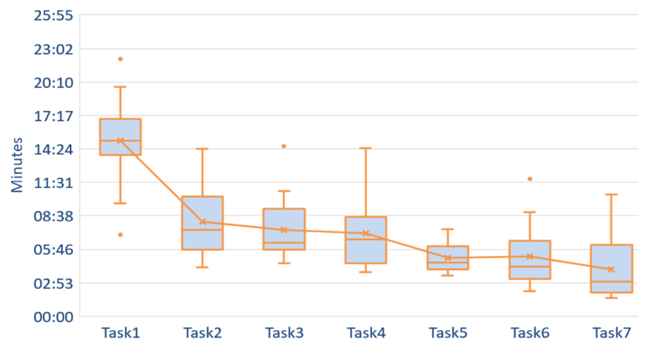


Figure 5: The average time to perform each task.

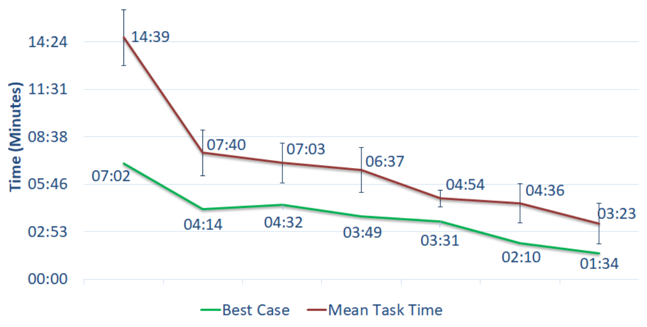


Figure 6: Mean time per task vs. best achieved time per task.

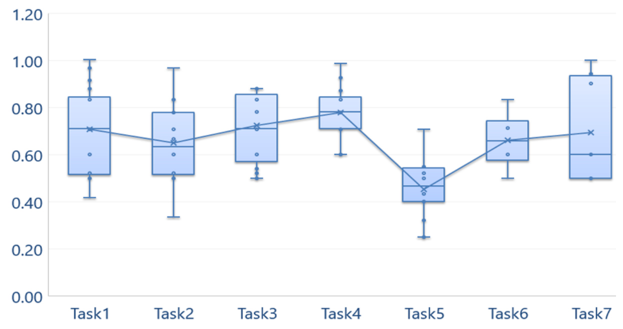


Figure 7: The average lostness score each task.

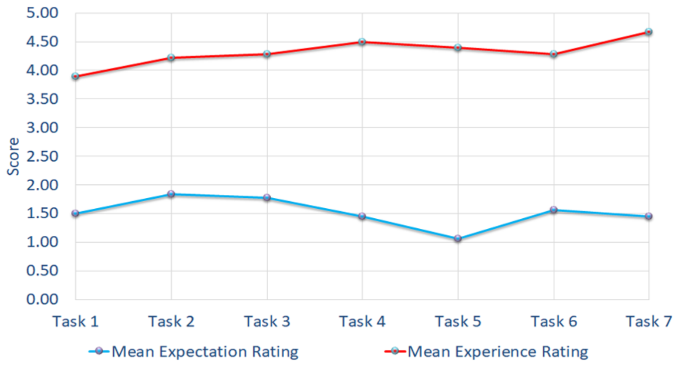


Figure 8: Pre/post-task ratings for each pre-study challenge.

assumed that the tool would notify them of errors. However, most of the errors were not automatically detected and reported by the tools' consistency checking until the subjects asked for it. It is possible that subjects thought their solutions were correct and did not invoke the tools' consistency checking.

6.2 Efficiency (RQ.2)

Efficiency was investigated by means of two metrics: time on tasks, and lostness. Fig. 5 depicts the results of the time that the subjects took for the tasks. To ensure that our results are meaningful, the times for the Incomplete tasks are not included in our analysis. This is because an unsuccessful subject could take a very long time to give up or to be asked to move on the next task, thus it can dramatically raise the average time on the tasks.

Tullis and Albert [4] suggest that one way to assess time on task is to compare the average time it took all subjects to perform a particular task with the minimum time it took to perform that task. Fig. 6 shows that the subjects' average times on most of the tasks were almost twice that of the best achieved times on tasks.

Worse, even the most-efficient subject was not as efficient as he/she could have been as evidenced by the lostness scores. Fig. 7 shows the average lostness scores for subjects' Generally Complete tasks. The Incomplete tasks are not included in the lostness scores because subjects were not "lost" when they left tasks Incomplete-They presumed they had met the task's requirements.

Using Smith's [45] threshold for lostness scores of 0.4, we see that, on average, the only task that the subjects could perform with a fairly acceptable lostness score was Task 5. In all other tasks, the subjects exceeded the acceptable lostness score which suggests the tools' interfaces are inefficient.

6.3 Satisfiability of Users' Expectations (RQ.3)

We answer RQ.3 by comparing Pre-Task Expectations against Post-Task Experiences. The result indicates that the subjects expected the tasks to be easy (based on their understanding of the task and the tools). However, their Post-Task Experience Ratings show that the tools did not meet their expectations. Fig. 8 shows the gap between the subjects' expectations and experiences in how difficult it was to use the tools to perform the tasks. The subjects, on average, expected the tools to ease modelling challenges (i.e., the mean Likert score was well below the neutral level of 4), but the subjects' experience ratings leaned towards dissatisfaction (i.e., the mean Likert score was slightly above the neutral value). This suggests that tools are not meeting the users' expectations on alleviating modelling tasks. Note that although the subjects could presumably learn from previously performed tasks, the subjects post-experience scores suggest that the tasks became increasingly harder for them.

6.4 Users' Satisfaction (RQ.4)

We used CSUQ to measure the satisfiability and usability of the tools under test where subjects answered questions about: 1) Overall Satisfiability (OVERALL), 2) System Usefulness (SYSUSE), 3) Information Quality (INFOQUAL), and 4) Interface Quality (INTERQUAL). Subjects specified their level of agreement based on the Likert scale that ranged from 1 (Strongly Disagree) to 7 (Strongly

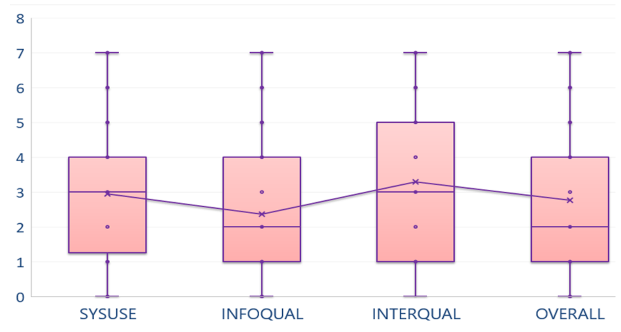


Figure 9: Box-plot based on the result from CSUQ.

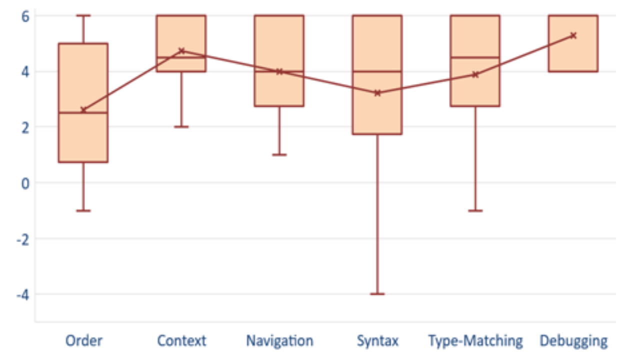


Figure 10: Mean discrepancy between the pre- and post-session ratings for each pre-study challenge.

Agree) with the neutral value of 4. Fig. 9 shows that subjects were slightly dissatisfied with the tools’ usability (SYSUSE) and the interface quality (INTERQUAL) (mean value of ~3), and were more strongly dissatisfied with the tools’ ability to provide the relevant or the contextual information during the tasks (mean value of around 2.5 for the information quality (INFOQUAL)).

6.5 The Most-Severe Challenges (RQ.5)

We investigated the users’ most-severe challenges by means of three different techniques: 1) Pre- and Post-Session Ratings, 2) Behavioural Metrics, and 3) Analysing Errors. The three analyses produced the same results.

6.5.1 Pre- and Post-Session Ratings. We used Pre- and Post-Session Ratings to identify pre-study challenges that users expected the tools to alleviate, but that the tools did not. Fig. 10 depicts the mean of the differences between Pre-Session Expectation and Post-Session Experience ratings with respect to the pre-study challenges. The figure shows that the subjects’ expectations and experiences had the greatest disparity with respect to the *Context* and *Debugging* challenges. That is, the subjects expected to face the least difficulty regarding the two challenges, but instead experienced the most difficulty. Moreover, the short box plots for these two challenges indicate a high degree of agreement among the subjects’ views. These results suggest that tool providers should propose tool advances that address these two prominent challenges over other tool advances.

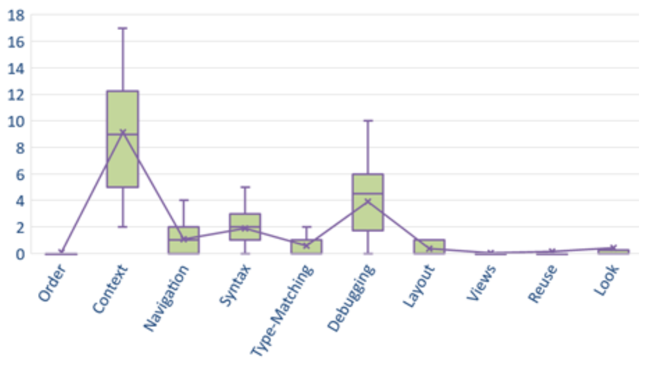


Figure 11: Mean frequency of each challenge expressed verbally by each subject.

6.5.2 Behavioural Metrics. The think-aloud protocol helped us identifying the subjects’ cognitive difficulties and challenges. We started with a list of the categories of challenges based on the six pre-study challenges. This list was then extended by four more categories as we learned more from the verbal analysis. The four new categories are: 1) ***Layout***: issues related to the layout of the diagrams, 2) ***Views***: issues related to the viewing mechanisms to combine different diagrams, 3) ***Reuse***: issues faced when reusing (i.e., copy and pasting) model elements, and 4) ***Look***: appearance-related dissatisfactions (e.g., shapes and colors).

By analysing the subjects’ verbal expressions, we were able to correlate each expression to its pertinent category of challenge(s). Sometimes, a statement could fall in two or more categories. In such a case, we correlated the statement to all of the applicable categories. The result of our verbal analysis is shown in Fig. 11. The figure indicates the two major challenges for the users were *Context* and *Debugging*. Some of the statements that the subjects made for *Context* were: "[While trying to remember the name of the elements] The tool should give me some recommendation." ,"what was the name of the class!", and "Oh! I forgot the name again...".

6.5.3 Analysing Errors. The errors rates presented in Fig. 4 show that almost all the subjects introduced inconsistencies to the model during the four model-editing tasks. Further analysis showed that the majority of these inconsistencies were related to referring to an incorrect or an undefined element. We believe that this relates to the *Context* challenge and that the subjects had difficulties recalling the intended model elements in the Class diagram. The tasks 5, 6, and 7 were designed to gauge the severity of the *Debugging* challenges, and the results presented in Fig. 3 show a high degree of failure in the subjects’ average performance for these tasks.

Based on the above results, one can conclude that the *Context* and *Debugging* challenges are the most-severe challenges. Due to the space limitations, we did not present the data collected from other sources that could confirm the above results such as the frequency with which subjects switched to the Class diagram during tasks as a reference metric for the severity of the *Context* challenge.

7 DISCUSSION

Below are additional observations about the subjects’ behaviours, and attitudes about the tools, collected during the subjects’ sessions with the tools or through an open-ended textual feedback that was answered at the end of the study.

- Sometimes, the subjects were satisfied with how they fulfilled tasks and did not realize when they had created an inconsistent or erroneous model. Subjects expected to be warned about errors as they were being made but the tools report errors only when the user proactively invokes error-checking features. It is important that the tools have features that either prevent such inconsistencies and errors during model editing or report them during commission. Please note that model correctness pertains to whether the model express the same information as the experiment's textual description (i.e., correctness errors are not methodological errors).
- Some tools such as MagicDraw [18] or Visual Paradigm [36] allow users to cross-link the operations in the event/effects of a transition to related operations in the Class diagram. However, users avoid creating these cross-links and instead just set the name or label of the transition. It is unknown whether the subjects simply do not care or if they think that it will be complicated for them to cross-link the operations to the related attributes for every transition. Perhaps, the ideal would be if the tools allowed the user to type a transition as a text, and the tool could automatically cross-link.
- Some tools (e.g., UMLet) are very lightweight and easy to learn, and are useful for creating simple models but are not suitable for editing complex models because of the lack of features such as syntax-highlighting and auto-completion. In contrast, tools with these features often feel very heavyweight and complicated. As a result, the latter tools have a learning curve that intimidates users because of the many views and features that it offers without proper organization.

8 THREATS TO VALIDITY

This section discusses the threats to the validity of the study.

8.1 Construct Validity

As mentioned, we used instrumentation to collect data about the time spent on tasks. The instrumentation improves the level of accuracy of the collected timings; however, because we used the think-aloud protocol, it is probable that the thinking out load and our prompts to the user increased the actual time to solve tasks. Also, our measurements for the time includes the time that the subjects took to read the task description and answer the Pre- and Post-Task Ratings, although this is negligible.

It is possible that the worst performance in the last two tasks may be attributed to fatigue or to the paying policy ($20 for about 90 minutes, leave as soon as you finish), and not due to the task type. We tried to mitigate this threat by providing an informal atmosphere and allowing the subjects to take a break whenever they felt uncomfortable. We believe such fatigue cannot be avoided and is generalizable to a real working environment. In our opinion, not being able to locate an error in a model after around three minutes (see Fig. 6) suggests that the tools do not provide enough useful and usable features for such debugging tasks.

8.2 Internal Validity

One threat to internal validity was the subjects' familiarity with the UML. We tried to mitigate the risks by recruiting subjects who could passed our UML exercises. We also made sure that the subjects had previously passed at least one course that included UML modelling.

A related issue was that we asked the subjects' to self-declare their expectation of the tools' abilities and their proficiency with using the tools. Their answers might be influenced by how self-confident they are in general, and may not necessarily reflect their real UML proficiency. As a result, their modelling errors and instances of lostness might be caused not only by inadequate tool support, but also by the level of competence in UML modelling.

8.3 External Validity

The main threat to external validity is that the subjects were students rather than experienced modellers who work in industry. To mitigate this threat we kept the scope of the study and the size of the model quite small compared to industrial models of software systems. Moreover, the tasks were relatively small and easy to do in terms of size, complexity and duration. Nevertheless, we cannot rule out the possibility that the observed effects could have been different if the systems and tasks had been larger.

One other threat was the subjects' familiarity with the system being modeled. It is possible that the results would differ if the users had dealt with the model for a longer period of time. A related threat is that the modellers might have performed better if they had created the Class diagram themselves as then the modellers would have been more familiar with the model.

9 CONCLUSION

We conducted a formative user study to understand the difficulties and challenges of modellers when editing and debugging static and dynamic UML models as exemplified by Class and State-Machine diagrams. Our analysis on the subjects' performance, verbal expressions, Pre- and Post-Session Ratings, and errors in the tasks determined that the most-prominent challenges of modellers are: 1) remembering contextual information (*Context*) and 2) locating, understanding and resolving errors in models (*Debugging*).

Furthermore, the average discrepancy between the Pre-Task Expectation and Post-Task Experience Ratings showed that there is a notable gap between the users' expectations and the tools' capability to satisfy the expectations. Similarly, the result of the CSUQ indicated that the subjects' opinion on the tools' usability leaned towards dissatisfaction.

Next steps are to identify enhancements to tools that address the most-critical challenges. For each challenge, we will identify relevant human-cognition factors that might effectively reduce the challenge, and devise enhancements to the tools that reinforce the identified factors. We will also hold empirical user studies to assess the impact of the tool advances on modellers' effectiveness.

10 ACKNOWLEDGEMENT

We thank the 18 participants of our empirical user study. This work was funded by the Natural Sciences and Engineering Research Council (NSERC) of Canada and the University of Waterloo.

REFERENCES

[1] Silvia Abrahão, Francis Bourdeleau, Betty Cheng, Sahar Kokaly, Richard Paige, Harald Stöerrle, and Jon Whittle. 2017. User experience for model-driven engineering: Challenges and future directions. In *20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 229–236.

[2] Luciane TW Agner and Timothy C. Lethbridge. 2017. A survey of tool use in modeling education. In *20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 303–311.

[3] William Albert and E. Dixon. 2003. Is this what you expected? The use of expectation measures in usability testing. In *Proceedings of the Usability Professionals Association 2003 Conference, Scottsdale, AZ.*

[4] William Albert and Thomas Tullis. 2013. *Measuring the user experience: collecting, analyzing, and presenting usability metrics.* Newnes.

[5] Mohammad M. Alsuraihi and Dimitris I. Rigas. 2007. How effective is it to design by voice?. In *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...But Not As We Know It - Volume 2*. BCS Learning & Development Ltd., 159–162.

[6] Kathy Baxter, Catherine Courage, and Kelly Caine. 2015. *Understanding your users: a practical guide to user research methods.* Morgan Kaufmann.

[7] John Brooke. 1996. SUS: A quick and dirty usability scale. *Usability evaluation in industry* 189 (1996).

[8] J. P. Chin, V. A. Diehl, and L. K. Norman. 1988. Development of an instrument measuring user satisfaction of the human-computer interface. In *Proceedings of the Special Interest Group on Computer-Human Interaction (SIGCHI) conference on Human factors in computing systems - CHI '88*. ACM Press, New York, New York, USA, 213–218.

[9] Andrej Dyck, Andreas Ganser, and Horst Lichter. 2013. Model recommenders for command-enabled editors. In *International Workshop on Model-driven Engineering By Example (MDEBE)*. 12–21.

[10] Alexander Egyed. 2006. Instant Consistency Checking for the UML. In *Proceedings of the 28th International Conference on Software Engineering*. ACM Press, New York, New York, USA, 381–390. https://doi.org/10.1145/1134285.1134339

[11] John Erickson and Keng Siau. 2007. Can UML be simplified? Practitioner use of UML in separate domains. In *proceedings of Evaluation and Modelling Methods for Systems Analysis and Development (EMMSAD)*. 89–98.

[12] Irit Hadar and Anna Zamansky. 2015. Cognitive factors in inconsistency management. In *23rd International Conference on Requirements Engineering(RE'15)*. IEEE, 226–229.

[13] Constance L. Heitmeyer, Ralph D. Jeffords, and Bruce G. Labaw. 1996. Automated consistency checking of requirements specifications. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 5, 3 (1996), 231–261.

[14] Anders Hessellund, Krzysztof Czarnecki, and Andrzej Wąsowski. 2007. Guided development with multiple domain-specific languages. In *International Conference on Model Driven Engineering Languages and Systems (MODELS)*. 46–60.

[15] James H Hill. 2011. Measuring and reducing modeling effort in domain-specific modeling languages with examples. In *18th International Conference and Workshops on Engineering of Computer Based Systems (ECBS)*. IEEE, 120–129.

[16] Ko-Hsun Huang, Nuno Jardim Nunes, Leonel Nobrega, Larry Constantine, and Monchu Chen. 2011. Hammering Models: Designing Usable Modeling Tools. In *13th International Conference on Human-Computer Interaction (INTERACT)*. Springer, 537–554.

[17] John Hutchinson, Jon Whittle, Mark Rouncefield, and Steinar Kristoffersen. 2011. Empirical assessment of MDE in industry. In *33rd International Conference on Software Engineering (ICSE)*. IEEE, 471–480.

[18] No Magic Inc. 2013. Magicdraw, UML. (2013).

[19] Stuart Kent. 2002. Model Driven Engineering. In *International Conference on Integrated Formal Methods*. Springer, 286–298.

[20] Mik Kersten. 2007. *Focusing knowledge work with task context.* Ph.D. Dissertation. University of British Columbia.

[21] Andrew J. Ko, Thomas D. Latoza, and Margaret M. Burnett. 2015. A practical guide to controlled experiments of software engineering tools with human participants. *Empirical Software Engineering* 20, 1 (2015), 110–141.

[22] Maria Kutar, Carol Britton, and Trevor Barker. 2002. A comparison of empirical study and cognitive dimensions analysis in the evaluation of UML diagrams. In *Proceedings of the 14th Workshop of the Psychology of Programming Interest Group (PPIG 14)*.

[23] Ludwik Kuzniarz, Miroslaw Staron, and Claes Wohlin. 2004. An empirical study on using stereotypes to improve understanding of UML models. In *Proceedings of 12th IEEE International Workshop on Program Comprehension*. IEEE, 14–23.

[24] Christian F. J. Lange and Michel R. V. Chaudron. 2004. An empirical assessment of completeness in UML designs. In *Proceedings of the 8th International Conference on Empirical Assessment in Software Engineering (EASE '04)*. IET, 111–121.

[25] Christian F. J. Lange, Michel R. V. Chaudron, and Johan Muskens. 2006. In practice: UML software architecture and design description. *IEEE software* 23, 2 (2006), 40–46.

[26] James R. Lewis. 1992. Psychometric evaluation of the post-study system usability questionnaire: The PSSUQ. 36, 16 (1992), 1259–1260.

[27] James R. Lewis. 1995. IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction* 7, 1 (1995), 57–78.

[28] Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of psychology* (1932).

[29] Arnold M. Lund. 2001. Measuring Usability with the USE Questionnaire 12 General Background. *Usability interface* 8, 2 (2001), 3–6.

[30] Parastoo Mohagheghi, Miguel A. Fernandez, Juan A. Martell, Mathias Fritzsche, and Wasif Gilani. 2008. MDE adoption in industry: challenges and success criteria. In *International Conference on Model Driven Engineering Languages and Systems (MODELS)*. Springer, 54–59.

[31] Andreas Muelder. 2011. Yakindu Statechart Modeling Tools.

[32] Gunter Mussbacher, Daniel Amyot, Ruth Breu, Jean-Michel Bruel, Betty HC Cheng, Philippe Collet, Benoit Combemale, Robert B. France, Rogardt Heldal, James Hill, et al. 2014. The relevance of model-driven engineering thirty years from now. (2014), 183–200.

[33] Christian Nentwich, Wolfgang Emmerich, Anthony Finkelstein, and Ernst Ellmer. 2003. Flexible consistency checking. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 12, 1 (jan 2003), 28–63. https://doi.org/10.1145/839268.839271

[34] Leonel Nóbrega, Nuno Jardim Nunes, and Helder Coelho. 2007. The meta sketch editor. In *Computer-Aided Design Of User Interfaces V*. Springer, 201–214.

[35] Object Management Group. 2015. OMG Unified Modeling Language TM (OMG UML), Superstructure v.2.3. *InformatikSpektrum* 21, March (2015), 758.

[36] Visual Paradigm. 2013. Visual paradigm for uml. *Visual Paradigm for UML-UML tool for software application development* (2013), 72.

[37] Tanumoy Pati, Dennis C. Feiock, and James H. Hill. 2012. Proactive modeling: auto-generating models from their semantics and constraints. In *Proceedings of the 2012 workshop on Domain-specific modeling*. ACM, 7–12.

[38] Tanumoy Pati, Sowmya Kolli, and James H. Hill. 2017. Proactive modeling: a new model intelligence technique. *Software & Systems Modeling* 16, 2 (2017), 499–521.

[39] Parsa Pourali and Joanne M. Atlee. 2018. *An Experimental Investigation on Understanding the Difficulties and Challenges of Software Modellers When Using Modelling Tools.* Technical Report CS-2018-03. David R. Cheriton School of Computer Science, University of Waterloo. 44 pages. https://uwaterloo.ca/computer-science/sites/ca.computer-science/files/uploads/files/cs-2018-03.pdf

[40] G Ramesh, T V Rajini Kanth, and A Ananda Rao. 2016. Extensible Real Time Software Design Inconsistency Checker : A Model Driven Approach. In *Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS)*, Vol. I. Hong Kong.

[41] James Reason. 1990. *Human error.* Cambridge university press.

[42] Gianna Reggio, Maurizio Leotta, and Filippo Ricca. 2014. Who Knows / Uses What of the UML : A Personal Opinion Survey. *ACM/IEEE 17th Intl. Conf. on Model Driven Engineering Languages and Systems* (2014), 149–165. https://doi.org/10.1007/978-3-319-11653-2_10

[43] Jason E. Robbins and David F. Redmiles. 2000. Cognitive support, UML adherence, and XMI interchange in Argo/UML. *Information and Software Technology* 42, 2 (2000), 79–89.

[44] Forrest Shull, Janice Singer, and Dag IK Sjøberg. 2008. *Guide to advanced empirical software engineering.* Vol. 93. Springer.

[45] Pauline A. Smith. 1996. Towards a practical measure of hypertext usability. *Interacting with computers* 8, 4 (1996), 365–381.

[46] Monique Snoeck, Cindy Michiels, and Guido Dedene. 2003. *Consistency by Construction: The Case of MERODE.* Springer Berlin Heidelberg, Berlin, Heidelberg, 105–117. https://doi.org/10.1007/978-3-540-39597-3_11

[47] Jon Whittle, John Hutchinson, Mark Rouncefield, Håkan Burden, and Rogardt Heldal. 2013. Industrial adoption of model-driven engineering: Are the tools really the problem? In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 8107 LNCS. Springer Berlin Heidelberg, 1–17. https://doi.org/10.1007/978-3-642-41533-3_1

[48] Dina Zayan, Michał Antkiewicz, and Krzysztof Czarnecki. 2014. Effects of using examples on structural model comprehension: a controlled experiment. In *Proceedings of the 36th International Conference on Software Engineering*. ACM, 955–966.