A User-Centric Approach to Improve the Quality of UML-like Modelling Tools and Reduce the Efforts of Modelling

by

Parsa Pourali

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2019

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Silvia Abrahão Associate Professor, Dept. of Computer Science, Universitat Politècnica de València
Supervisors:	Joanne M. Atlee Professor, Dept. of Computer Science, University of Waterloo Krzysztof Czarnecki Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Internal Member:	Derek Rayside Associate Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Internal Member:	Werner Dietl Assistant Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Internal-External Member:	Nancy Day Associate Professor, Dept. of Computer Science, University of Waterloo

Statement of Contributions

The body of this thesis is based on a combination of published works. Various sections are adapted from the following list of publications:

- Chapter 4 draws on a technical report and a published conference paper. The paper is co-authored by my supervisor (Dr. Atlee) and me. I developed the experimental methodology and design. I also carried out the lab experiments and collected and analyzed the experimental data. Dr. Atlee provided feedback and comments throughout this process. Dr. Atlee also assisted with the writing of the paper.
  - Parsa Pourali and Joanne M. Atlee. "An Empirical Investigation to Understand the Difficulties and Challenges of Software Modellers When Using Modelling Tools." *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, pages 224 - 234. ACM, 2018.
  - Parsa Pourali and Joanne M. Atlee. *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, 2018.
- Chapter 5 of this thesis draws on three papers. One of the papers has been written by me, and the other two papers were co-authored by me and my supervisor (Dr. Atlee). I developed and implemented the tooling techniques. Dr. Atlee reviewed and commented on the manuscript.
  - Parsa Pourali. "Tooling advances inspired to address observed challenges of developing UML-like models when using modelling tools." *Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, pages 168 - 173. ACM, 2018.
  - Parsa Pourali and Joanne M. Atlee. "A Focus+Context Approach to Alleviate Cognitive Challenges of Editing and Debugging UML Models." *Proceedings of the 22nd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, pages 183 - 193. ACM, 2019.
  - Parsa Pourali and Joanne M. Atlee. "UCAnDoModels: A Context-based Model Editor for Editing and Debugging UML Class and State-Machine Diagrams." *Proceedings of the 22nd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, pages 779 - 783. ACM, 2019.

- Chapter 6 has been incorporated within a conference paper that is co-authored by my supervisor (Dr. Atlee) and me. I developed the experimental methodology and design, and Dr. Atlee reviewed the design. I also carried out the lab experiments and collected and analyzed the experimental data. Dr. Atlee also assisted with the writing of the paper.
  - Parsa Pourali and Joanne M. Atlee. "A Focus+Context Approach to Alleviate Cognitive Challenges of Editing and Debugging UML Models." *Proceedings of the 22nd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, pages 183-193. ACM, 2019.

Abstract

As software systems grow in size and complexity, their development and maintenance are becoming increasingly challenging. Model-Driven Engineering (MDE) has been proposed as a means to increase the developer's productivity of such large-scale complex software systems. Despite its benefits, MDE has not been fully adopted in the software industry due to several barriers. Research has shown that modelling tools are amongst the top barriers for the industry's reluctance to adopt MDE, mostly because there is a little investigation of the modellers' interactions with modelling tools when editing and debugging models, which are cognitively difficult tasks. More specifically, MDE tool research has not considered 1) a thorough analysis of modellers and their tasks, to understand their challenges of using modelling tools, 2) the underlying human-cognitive factors, and 3) a systematic assessment of the effectiveness of proposed solutions (i.e., tooling techniques) on human users.

This thesis argues that MDE tools can be enhanced to overcome (some of) the challenges of adoption by considering human-cognitive factors (i.e., user-centric) when designing and proposing model-easing techniques for model editors. We advance our thesis in three main steps. As a first step, we conducted an empirical study to identify the most-severe cognitive difficulties of modellers when using UML model editors. In our study, we asked the recruited subjects to perform several model-editing and model-debugging tasks. We collected information during the sessions that could help us understand the subjects' cognitive challenges. The results show that users face multiple challenges, amongst which the most prominent challenges are remembering contextual information when performing a particular modelling task; and locating, understanding, and fixing errors in the models.

In the second step, we identified the cognitive factors that drive the most prominent challenges and subsequently devised several tooling advancements that provide enhanced cognitive support and automation in the users' interaction with a model editor. The philosophy behind our tooling advancements is to provide the contextual information that are relevant to performing a particular modelling task, thereby, alleviating the modellers' cognitive challenges of recollecting information from different diagrams. We also proposed an on-the-fly error-resolution technique that aims at resolving errors as they occur. We implemented our Eclipse-based model-editor and embedded our tooling techniques in the tool.

Lastly, we conducted two empirical studies to assess the effectiveness of our model-editor on human users. The *Context* study aimed at evaluating our tool's ability to reduce the challenges of remembering contextual information, whereas the *Debugging* study aimed at assessing our tool's ability to improve the users' experience of debugging models. Our results reveal that our interfaces help users 1) improve their ability to successfully fulfil their tasks, 2) avoid unnecessary context switches among diagrams, 3) produce more error-free models, 4) remember contextual information, and 5) reduce time on tasks.

Acknowledgements

Undertaking a PhD is probably one of the most life-changing experiences that one can ever have in their life. Without the support and guidance from the PhD advisors, this is almost impossible. I would like to express my heartiest gratitude to Prof. Joanne M. Atlee, who has always been a kind, friendly, and supportive PhD advisor for me. Her positive attitude and smile have always encouraged me to move forward and to overcome frustrations during my PhD. My achievements would not have been possible without her supportive, friendly, and professional supervision. I have learned a lot from her.

I would like to thank Prof. Krzysztof Czarnecki as my PhD co-supervisor. His patience and dedication to work helped me understand the significance of putting more efforts into the work that I am doing and wait for better results. I am also grateful to have Profs. Nancy Day, Derek Rayside, and Werner Dietl as my PhD committee members. I would like to thank them for their participation and constructive feedback that they have provided at different stages of my research.

Many thanks to the Faculty of Engineering of the University of Waterloo for providing me with opportunities and facilities to achieve my PhD, as well as realizing my ideas and developing my expertise. I would also like to acknowledge the support from the funding sources Natural Science and Engineering Research Council of Canada (NSERC) and University of Waterloo.

I think I will always remember the good days I had in Davis Center's lounge room discussing matters from research to politics with my friends. I am fortunate to have such wonderful friends.

The support and joy I received from my sister and brother-in-law, Mahsa and Mahyar, and their children, Hana and Hoonam, helped me to overcome the difficulties of a PhD life! Thanks for being there for me.

Words cannot express my special and deepest gratitude to my parents, Mousa and Marzieh. Their love and support is the most precious thing I have in my life. My dad is probably the happiest person in the world after me for fulfilling this PhD. I dedicate this thesis to my parents.

dedicated to my parents, Mousa and Marzieh...

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

"The power of the unaided mind is highly overrated. Without external aids, memory, thought, and reasoning are all constrained. But human intelligence is highly flexible and adaptive, superb at inventing procedures and objects that overcome its own limits. The real powers come from devising external aids that enhance cognitive abilities. How have we increased memory, thought, and reasoning? By the inventions of external aids: It is things that make us smart."

— – Donald Norman, *"Things That Make Us Smart"*

We have witnessed a significant increase in the size of software systems. These systems typically involve many human labours and last for several years [29]. Developing and maintaining such large-scale complex systems has always been challenging for software engineers [29]. As a solution, Model-Driven Engineering (MDE) addresses complexity by raising the level of abstraction in a software specification and facilitating the automation of code generation and software verification [48][71]. More specifically, MDE is a software design methodology that focuses on improving the productivity of software developers by allowing the software engineers to test the important properties of the system before coding begins. MDE involves several artefacts amongst which models are the core assets. The engineer designs *precise and analysable* models to express the elements of a software system and its properties (by means of graphical or textual notations) in order to enable automated code generation and improve understanding and reasoning of the system.

Despite their benefits, models have not been well-adopted in industry because of various barriers, amongst which the lack of proper tooling techniques is the most crucial [46][71].

The main reason that practitioners are reluctant to adopt models is that they often find it *cognitively difficult to create, edit, and debug* precise and analysable models [42][77][88] when using model editors. Researchers investigate proficient tools and methods to overcome the practitioners' cognitive challenges of using model editors [40][71][76], but the editors are not fully adopted because they are not meeting the users' expectations. More specifically, tool researchers mostly focus on proposing heuristic *artefact-centric* techniques that focus on language's syntax and semantics, such as automated support for detecting syntax- or semantic-related errors, or auto-completions that fulfill syntactic and semantic expectations. The solutions proposed by these approaches take into account *only* the meta-model, the state and properties of the model, and the constraints on them. Such artefact-centric approaches neither employ human-cognitive theories nor conduct empirical user studies (during the design process) to enhance the usability of their proposed solutions.

Although artefact-centric approaches offer many capabilities to improve the editors' usefulness, their effectiveness on human users is rarely assessed. Tool developers rarely investigate modellers' prominent challenges (by performing a thorough analysis of the modellers and their tasks), and they do not evaluate the effectiveness of modellers in using their tools. Thus, they miss opportunities to address *usability* concerns, leading to a chasm between how easy modellers expect a modelling task to be and how complex it is when using model editors. We argue that a useful model editor should also meet the usability requirements, which is mainly related to the *design decisions* and *cognitive supports* that reduce the cognitive load being imposed on the user as a consequence of using the editor.

To address the usability of MDE editors, MDE researchers have proposed new research directions [1] that investigate users' cognitive challenges and correlate the challenges with the cognitive theories to understand users' difficulties and improve users' experience (UX) of using model editors. They ask usability-related research questions such as: *How effective is the tool in alleviating users' tasks? How difficult is the tool to learn? How easy is the tool to use?* They have proposed *user-centric* techniques, which provide cognitive support to users by offering various features such as easy-to-use user interfaces (UIs), editors, and navigation features, thus improving users' visual and spatial capabilities, working memory, task-specific design understanding, etc. Moreover, systematic evaluation of the effectiveness of user-centric techniques has become a trend in recent years. It is important to note that user-centric techniques do not conflict with artefact-centric techniques. In fact, in most cases, they complement the artefact-centric techniques by addressing their usability aspect.

Although both artefact-centric and user-centric techniques have made significant improvements to the quality of MDE tools with respect to usefulness and usability issues, there is still a huge usability gap between what the tools offer and what the users expect. The reasons are, in part,

1. Tool designers have not identified and understood the difficulties and challenges of the users through empirical observations or field studies.

2. They have not taken into account modellers' behavioural profiles or human-cognition factors that can explain users' difficulties and challenges.

3. They have conducted little in the way of empirical evaluation of the effectiveness of modelling tools in supporting human users.

In this thesis, we aim to understand users' difficulties and augment users' cognitive abilities by proposing a balanced set of artefact-centric and user-centric techniques that, combined together, ease the tasks of *editing* and *debugging* models. Accordingly, our approach, which we call **U**ser-**C**entric and **A**rtefact-Ce**n**tric **D**evelopment **o**f **Models** (*UCAnDoModels*), employs and incorporates both user-centric and artefact-centric strategies to enhance the model editors' quality. The artefact-centric aspect of our approach includes backend-related model-editing and model-debugging mechanisms such as automation and real-time error resolution techniques, whereas the user-centric aspect of our work ensures that these techniques are designed and embedded in the tool in such a manner that they conform to human-cognition theories. Our approach consists of four important steps slightly adapted from [10] and [49] (see Fig. 1.1): 1) understanding the foremost difficulties experienced by model editor users, 2) correlating each of the difficulties to human-cognitive challenges, 3) proposing tooling solutions that address the cognitive challenges, and 4) assessing the effectiveness of the proposed solutions on human users. We believe that such an approach can enhance the quality of model editors and modellers' experience of using the editors.

Thesis statement:
Current MDE tools can be enhanced by employing artefact- and user-centric techniques to help reduce the effort of editing and debugging analysable models and improve users' experience of using model editors.

1.1 Thesis Context

The scope of our work is enhancements to model editors for feature-oriented models expressed in the Feature-Oriented Requirements Modelling Language (FORML)[97], which

Table 1.1: List of UML features (for Class and State-Machine diagrams) that are not supported by our model editor.

Diagram	Unsupported Features
Class Diagram	Stereotypes (e.g., Interface, Abstract)
	Association's Navigability
	N-ary Association
	Visibility of a Named Element (i.e., Public, Package, Protected, Private)
	Modifiers for Properties (e.g., readOnly, ordered)
	Dependency and Realization Relationships
State-Machine Diagram	Deep and Shallow History Pseudostates
	Join Pseudostate
	Fork Pseudostate
	Junction Pseudostate
	Choice Pseudostate
	EntryPoint Pseudostate
	ExitPoint Pseudostate
	Terminate Pseudostate
	State's Internal Behaviours (e.g., entry)
	Time Events (i.e., When, At)
	Protocol State-Machines

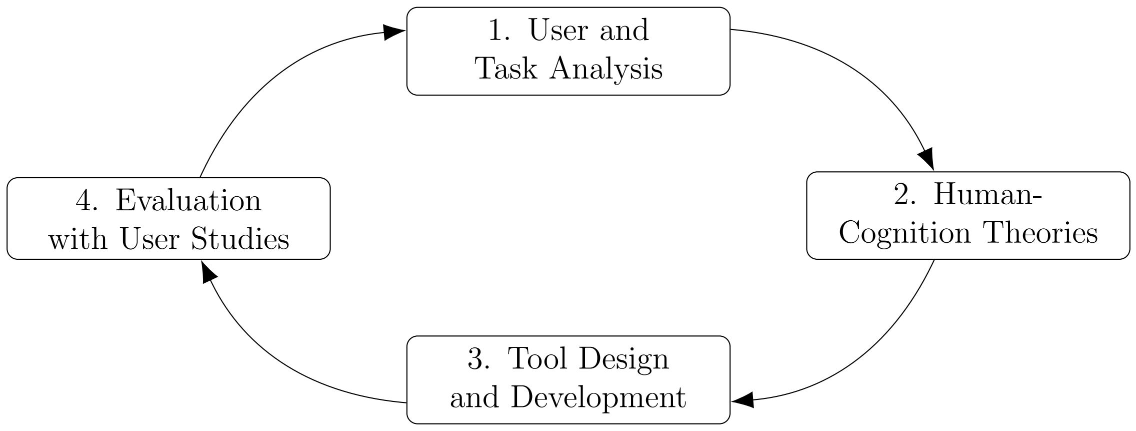

1. User and Task Analysis
2. Human-Cognition Theories
3. Tool Design and Development
4. Evaluation with User Studies

Figure 1.1: The four steps of our approach.

is a formal language that is expressive and amenable to *analyses*. Specifically, we focus on tooling techniques for creating and editing FORML diagrams: Feature Model diagrams, Feature Class diagrams, and Feature Module diagrams (which are enhanced State-Machine diagrams). Because FORML notations and diagrams are based on the Unified Modelling Language (UML), the context of our proposed enhancements also includes UML model editors, which have a broader user base. That is, our model editor allows UML modellers to develop *precise and analysable* UML Class and State-Machine diagrams. However, for the sake of this thesis, we implemented the most important and prevalent features of UML Class and State-Machine diagrams (see Table 1.1 for a list of UML features that are not supported by our tool). Although our enhancements are devised in the context of Feature-Oriented Modelling, our model editor is only evaluated with respect to the capabilities and usability of UML tools created by tool vendors and modelling researchers. More specifically, the scope of the thesis can be considered as follows:

- The scopes of the empirical user studies conducted in the thesis discussed in Chapters 4 and 6 are bounded to UML Class and State-Machine diagrams to extend the generalizability of our findings. The target population of users for these user studies are UML modellers who use model editors to edit and debug UML Class and State-Machine diagrams. These users can be either novice or experienced modellers with a level of competency in UML.
- The scope of the proposed solutions discussed in Chapter 5 is FORML diagrams. However, the proposed solutions are also applicable to UML Class and State-Machine diagrams because of the high similarity between FORML and UML syntax and se-

mantics. Moreover, the users of the proposed solutions are meant to be UML modellers who have a minimum level of knowledge about UML and UML tools. The proficiency of a user (i.e., novice or experienced) is not deemed to affect the applicability of the proposed solutions.

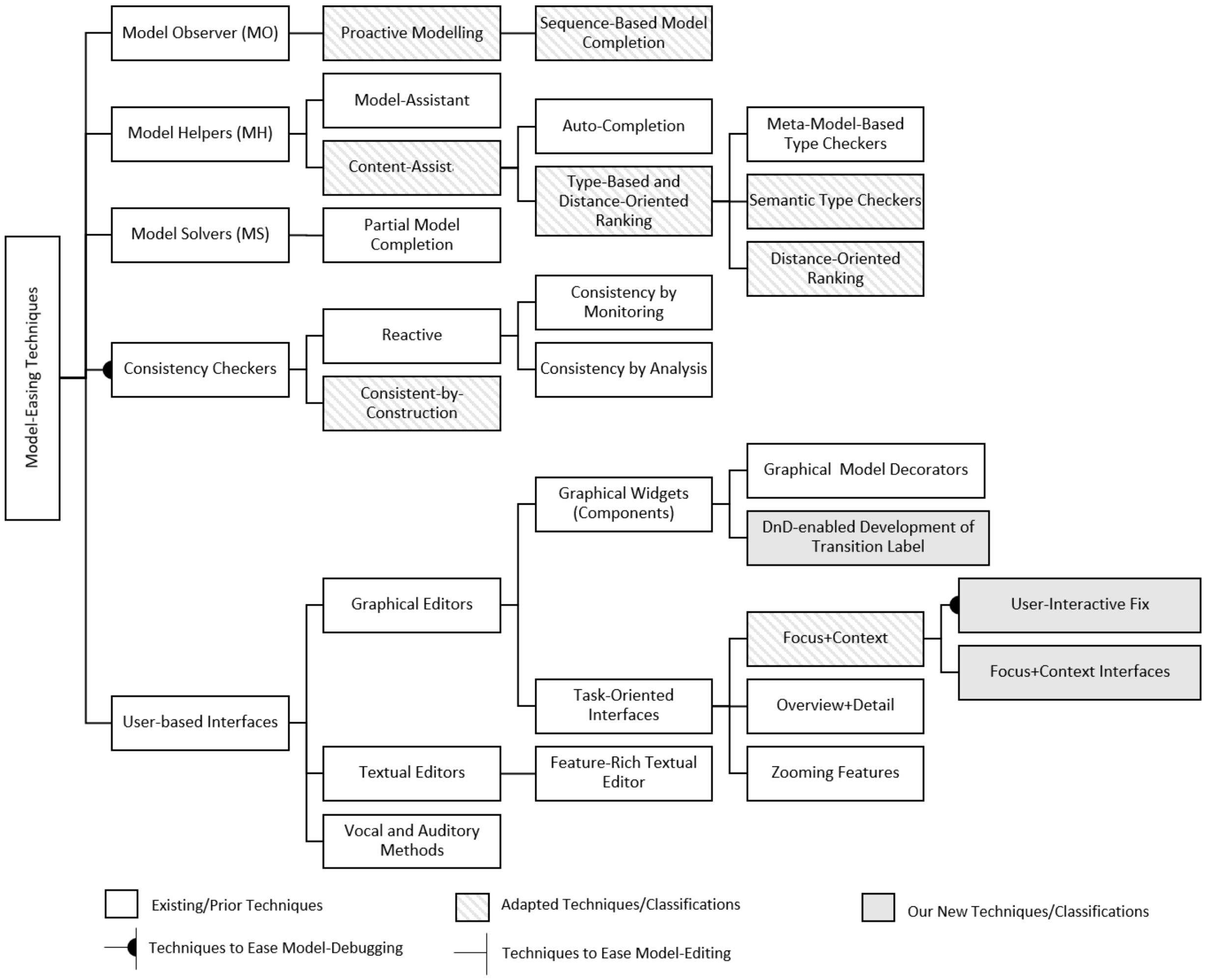
1.2 Thesis Contributions

Our research provides insight into how modellers interact with model editors and how cognition psychology theories can be exploited to improve this interaction. The overall contributions of our work include:

- *Analysing modellers and their tasks to learn about the prominent difficulties and challenges that they face when using UML model editors.* We conducted a two-phase user study in which we investigated modellers' usage of UML model editors, including the tools' effectiveness, users' efficiency, users' satisfaction, the gap between users' expectations and experiences, and users' cognitive difficulties. The subjects were asked to perform four model-editing and three model-debugging tasks. The results show a substantial gap between users' expectations of tools' abilities to alleviate the challenges and the users' actual experiences of using the tools. The results also reveal that *remembering contextual information* and *identifying and fixing errors and inconsistencies* are the most-prevalent challenges and are most in need of consideration from tool vendors. Tool researchers can learn from our work and redirect their research to propose solutions that can alleviate the identified challenges.

- *Devising tooling advancements such as Focus+Context [21] interfaces and increased automation that aim to reduce the identified cognitive challenges of developing models.* Our techniques provide users with the information (*Context*) that are relevant to performing a particular task (*Focus*). The outcome is a semi-formal (FORML-based) UML model editor that facilitates many of the modellers' tasks by employing our proposed interfaces and automation features. Moreover, we propose on-the-fly error resolution techniques that allow the modeller to locate, understand, and fix model inconsistencies in real-time without being overly disrupted from their modelling task.

  Fig. 1.2 shows a list of model-easing techniques and their corresponding categories. Also, it shows an overview of our techniques and how they relate to existing techniques in the literature. Some techniques ease model-debugging (boxes that are connected by a filled-arc), whereas other techniques ease model-editing (boxes that



Model-Easing Techniques
Model Observer (MO)
Proactive Modelling
Sequence-Based Model Completion
Model Helpers (MH)
Model-Assistant
Content-Assist
Auto-Completion
Type-Based and Distance-Oriented Ranking
Meta-Model-Based Type Checkers
Semantic Type Checkers
Distance-Oriented Ranking
Model Solvers (MS)
Partial Model Completion
Consistency Checkers
Reactive
Consistent-by-Construction
Consistency by Monitoring
Consistency by Analysis
User-based Interfaces
Graphical Editors
Graphical Widgets (Components)
Graphical Model Decorators
DnD-enabled Development of Transition Label
Task-Oriented Interfaces
Focus+Context
User-Interactive Fix
Focus+Context Interfaces
Overview+Detail
Zooming Features
Textual Editors
Feature-Rich Textual Editor
Vocal and Auditory Methods
Existing/Prior Techniques
Adapted Techniques/Classifications
Our New Techniques/Classifications
Techniques to Ease Model-Debugging
Techniques to Ease Model-Editing

Figure 1.2: Feature tree for model-easing techniques.

are connected by regular lines). The feature tree differentiates between existing techniques (non-filled boxes with solid line borders), techniques that we adapt (gray-striped boxes), and novel techniques proposed by us (grey-filled boxes).

More specifically, we introduce new Focus+Context Editors that aim to reduce the effort to remember contextual information when performing a particular modelling task, by displaying the relevant contextual information in an integrated interface. The Focus+Context interfaces [21] also allow the modeller to modify the contextual information, if needed. We also propose tooling enhancements that alleviate the difficulties of creating Correct-by-Construction models (which are *precise* and *analysable*) by employing a User-Interactive Consistency Management interface. Moreover, we adapt some of the existing tooling techniques and propose enhancements to them. For instance, we employ a distance-oriented method to order and rank the model elements to be presented to users when developing transition labels. Also, our model editor embeds a novel filtering mechanism which takes a proactive approach and filters out those model elements that are not sound by type. We discuss the description of the existing techniques and the details of our enhancements in Chapters 3 and 5, respectively.

In addition, we provide an integrated environment for feature-oriented modelling; currently, users must use different tools to develop different aspects of a feature-oriented model (e.g., one tool to develop Feature Modules and another tool to develop a Class diagram).

- *Evaluating the effectiveness of our proposed tooling enhancements by conducting two user studies.* The results of the user studies show that our model editor is significantly more effective than other model editors in alleviating modellers' cognitive challenges. More specifically, our results reveal that our interfaces help users to: 1) improve their ability to successfully fulfil modelling tasks, 2) avoid unnecessary context switches among diagrams, 3) produce more error-free models, 4) remember contextual information, and 5) reduce time on tasks. We also provide empirical implications for tool vendors and researchers to enhance and improve the quality of model editors, which is crucial for a greater adoption of MDE by industry.

1.3 Overview of Dissertation Contents

The rest of the dissertation is structured as follows:

Chapter 2 describes the background information relevant to the thesis such as fundamentals of the used modelling paradigms (i.e., FORML and UML), the technologies that were used to implement our tool, and the fundamentals of the cognitive theories that are most-relevant to our thesis.

Chapter 3 surveys the related work that has been done and discusses how our approach contributes to the state-of-the-art.

Chapter 4 explains the formative user studies that we conducted to identify and understand the prominent challenges that modellers face when using model editors. It presents the experimental design as well as the results obtained from the user studies. This chapter draws on a technical report [83] and a published conference paper [80].

Chapter 5 presents our UCAnDoModels approach which includes our proposed tooling advancements to alleviate the identified cognitive challenges. It also correlates our tooling advancements to the relevant cognitive abilities. This chapter draws on three conference papers [79][81][82].

Chapter 6 shows how we evaluated our UCAnDoModels techniques. We demonstrate that our techniques can effectively reduce the challenges of modellers when using model editors. It also presents the design of our empirical studies and their results. This chapter has been incorporated within a conference paper [81].

Chapter 7 concludes the dissertation and highlights the future research directions.

Chapter 2

Background

"In debugging a program, in writing a paper, in doing financial analysis of a firm, in attempting to reason about a machine, limitations on the number of mental things that can be kept track of lay a strong constraint on human cognitive capabilities."

— – Card et al., *"Window-based Computer Dialogues"* [18]

This chapter covers the relevant background that is needed to understand the rest of this thesis, including the Unified Modelling Language (UML), Feature-Oriented Requirements Modelling Language (FORML), Eclipse modelling technologies and frameworks, and the human-cognition theories that are important to understand the rationale of our tooling techniques.

2.1 Unified Modelling Language

Among all the artefacts that Model-Driven Engineering (MDE) entails, models are the primary ones, as opposed to other artefacts such as codes that are later generated from models. Models can be developed in various modelling languages. Some companies use their own Domain-Specific Language (DSL) whereas others prefer to use a more general-purpose language such as the UML or Systems Modelling Language (SysML). However, the majority of the models that are developed are based on the UML or UML-like notations as the UML has become the de-facto standard for modelling software systems [54] and is actively taught and used in academia.

The UML consists of several types of diagrams which can be classified into *static* and *dynamic* diagrams [74]. Static diagrams (e.g., Class diagrams and Object diagrams) show the structural constructs of a system and the relationships among the entities and objects, whereas the dynamic diagrams (e.g., Sequence diagrams and State-Machine diagrams) are used to model behavioural aspects of the system [74] The focus of our thesis is related to the most prominent UML static and dynamic diagrams: Class diagrams and State-Machine diagrams.

A Class diagram represents a system's structure, entities, and their properties (i.e., attributes and operations) as well as the relationships among entities [104]. Real-world entities such as physical objects and concepts are described in terms of *classes* [104]; and the functionalities and persistent states of an entity are represented by the operations and attributes of the corresponding class [104]. The relationships among classes can be specified as *Association*, *Aggregation*, *Composition*, or *Generalization*. An association means that there is a link among the instances of the associated classes. Composition and aggregation relationships represent *whole-part* relationships, whereas generalization represents an *is-a* or *is-a-kind-of* relationship (also known as *inheritance*) [104]. Fig. 2.1 shows an example Class diagram consisting of a few classes. It shows the attributes and operations of the classes (e.g., *car_at_gate* and *isGateBlocked* for the *Sensor* class) and illustrates how these classes are related to each other through edges. An edge can represent an association (e.g., a *Transponder* can be sensed by a *Sensor*), a composition (e.g., a *Gate* includes *Sensors*), or a generalization (e.g., *GateA* class inherits the properties of the *Gate* class).

For each class in the Class diagram, there can be a State-Machine diagram that represents its behaviour. The state that a new object will be in at the time of its creation is referred to as its *initial state*, and states that the object will be in discrete segments of time are called *(simple) states* [104]. Additionally, the UML defines *composite states* as a state that contains one or more sequential (disjoint) or concurrent (orthogonal) *regions*, each of which aggregates a set of sub-states [104].

A *transition* links one state to another succeeding state, and is stimulated by a triggering event. The UML specification defines the standard format for writing a transition expression as ***event[guard]/actions***[104]. An *event* represents the operations that stimulate the object to transit from one state to another state. Ideally, an event refers to the operations that are defined in the current class or in the classes that have a relationship to the current class [104]. A *guard* is a Boolean expression that can be used to strengthen a triggering event: the guard must also hold for the transition to execute. The functional

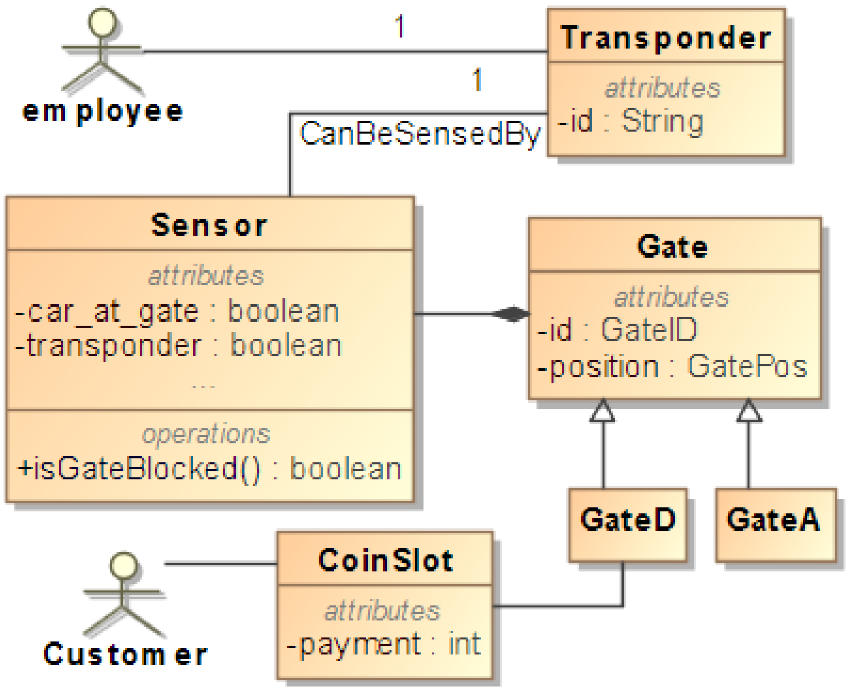


Figure 2.1: An example of a Class diagram.

responses (e.g., changes to phenomena in the environment) of an object after executing a transition are represented as the *actions* of the transition [104].

Fig. 2.2 shows an example State-Machine diagram for the class *GateA* from Fig. 2.1. It consists of an initial pseudo-state and two simple states (i.e., *Closed* and *Open* states) represented by a black filled circle and rectangles, respectively. The *Closed* and *Open* states are connected by *transition* links. For instance, the transition from the *Open* state to the *Closed* state expresses that *after five seconds* the entity (i.e., *GateA*) transitions to its *Open* state if the gate is not *blocked* and that the gate's *position* changes to *Down*.

2.2 Feature-Oriented Requirements Modelling Language

A *requirement* is an objective, a capability, or a functionality of a software system that needs to be maintained or achieved within the context of its environment (i.e., real-world) [24]. Feature-Oriented Software Development (FOSD) is a development paradigm that

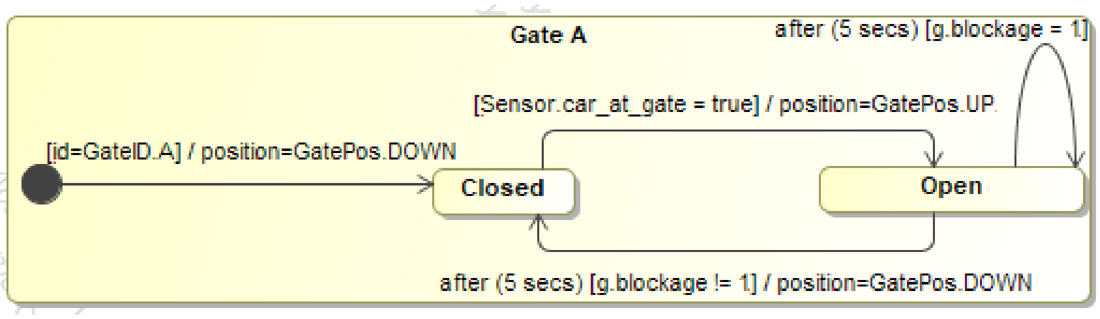

Gate A
after (5 secs) [g.blockage = 1]
[Sensor.car_at_gate = true] / position=GatePos.UP
[id=GateID.A] / position=GatePos.DOWN
Closed
Open
after (5 secs) [g.blockage != 1] / position=GatePos.DOWN

Figure 2.2: An example of a State-Machine diagram.

abstracts a software system's requirements as a composition of coherent and identifiable bundles of functionalities referred to as *features* (e.g., Cruise Control in an automobile) [97]. FOSD increases the degree of modularity of a software system as well as improving its construction and evolution by allowing the features to be developed in isolation and by different vendors [97]. Feature orientation is more desirable also from the stakeholders' (e.g., customers) point of view because a product can be better understood in terms of their features [97]. It is mostly relevant in Software Product Lines (SPLs), which are software-intensive systems that are constructed from a common set of features and share similar functionalities [97].

A suitable (formal) language for designing feature-oriented models is the Feature-Oriented Requirements Modelling Language (FORML), which expresses the requirements in terms of the effects they have on the product's environment [97]. It provides a formal and precise syntax and semantics to enable formal analyses of the FORML models [96]. To promote adoption by practitioners, FORML offers Feature Class and Feature Module diagrams based on the notations of UML Class and State-Machine diagrams, respectively. In addition, FORML offers a Feature Model diagram to illustrate a system's set of features, the relationships among features, and the set of valid feature combinations (called *configurations*) of a SPL product.

2.2.1 Feature Model Diagram

FORML offers a Feature Model that represents the hierarchical structure of a system's set of features and their relationships. The root node of the hierarchy represents the system itself, and the child nodes represent the features that make up the system's possible units of functionality [97]. A feature can be mandatory (connected by a filled diamond ⟵) or optional (connected by an empty diamond ◇–) if its parent feature is present in the system. Moreover, there may be constraints among the child features of a parent feature [97]:

- AND (▲) relationship: is attached to a feature and means that all of the child features must be present in a system if the parent feature is present,
- OR (⊞) relationship: is attached to a feature and means that at least one of the child features must be present in a system if the parent feature is present, and
- XOR relationship (⊠) relationship: is attached to a feature and means that exactly one of the child features must be present in a system if the parent feature is present.

As a running example, we use a parking lot SPL. A possible configuration (i.e., composition of several features) of the parking lot SPL is *GatedLot* (product), which refers to a gated parking lot. The *GatedLot* system can have two different types of gates: 1) Payable Gates, which allow customers to enter and pay for surface-level parking, and 2) Transponder-Enabled Gates, through which only authorized vehicles with a valid transponder can enter into the underground parking lot. A payable gate also allows a customer to pay in cash (i.e., coin, paper money, or tokens) or pay by card (i.e., a credit card or a prepaid membership card). For a transponder-enabled gate, a customer can be an employee of the complex or an authorized customer who paid for a transponder to benefit from the underground parking lot.

Fig. 2.3 shows a partial Feature Model for the *GatedLot* system. As can be seen, the root of the Feature Model is the *GatedLot* system, which includes *PayableGate* and *TransponderEnabledGate* features. The filled circles (or diamonds in our tool) for the *CoinPay* and the *PaperPay* features denote that these features are mandatory for the payable gates, meaning that the gate must accept coin and/or paper money as its payment method, whereas the empty circle for the *CardPay* feature shows that a payable gate may or may not accept cards to receive payments; however, if it is designed to accept a card payment method, it must at least accept payment by Visa or Master cards. Moreover, if the system includes the *Transponder-Enabled Gate* feature, then it must support mechanisms to allow both employees **and** subscribed transponder holders.

2.2.2 Feature Class Diagram

Similar to UML Class diagram, a Feature Class diagram represents the classes (i.e., entities) that exist in the environment of any product in the SPL, and the relationship among the

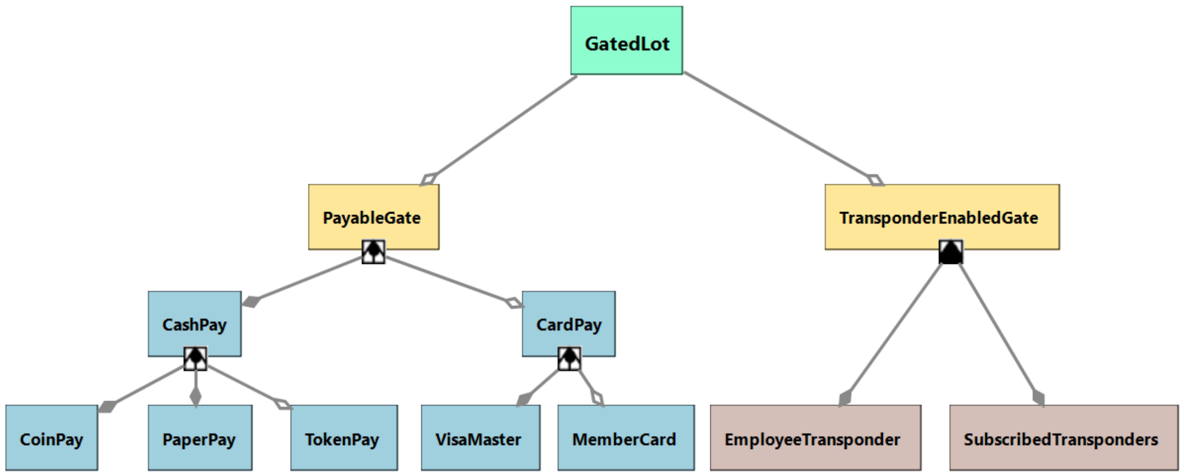


Figure 2.3: An example of a Feature Model for the GatedLot SPL.

entities. In addition, a Feature Class diagram includes *feature classes* that are mapped to features in the Feature Model. Every feature class in the Feature Class diagram has a set of *attributes* or *messages* (known as *operations* in UML). In particular, a feature can communicate to the product's environment (i.e., real-world entities) through its *input* and *output* messages [96]. Through an *input message*, environment agents can communicate to the feature, whereas through an *output message* a feature communicates to agents in the environment [96]. Fig. 2.4 presents a partial Feature Class diagram for the *GatedLot* system discussed above. It shows the real-world entities (e.g., employee) as well as the feature classes, the latter distinguished by a dashed border and a light brown background (e.g., CardPay). It also shows the input and output messages for the feature classes. For instance, the *SubscribedTransponder* feature, which is responsible for allowing customers who have subscribed to transponders to access the underground lot, senses from the environment if a car is approaching through a *setCarApproaching* input message. In the case that a car has a valid transponder, the feature directs to the environment to open the gate by issuing an *openGate* output message.

One can notice that a Feature Class diagram is an ontology of several *concepts* such as *classes*, *features*, and *messages*, which are expressed in UML Class diagram notations [96]. Moreover, a *concept* can be realized by one or more than one instances referred to as *objects* [96].

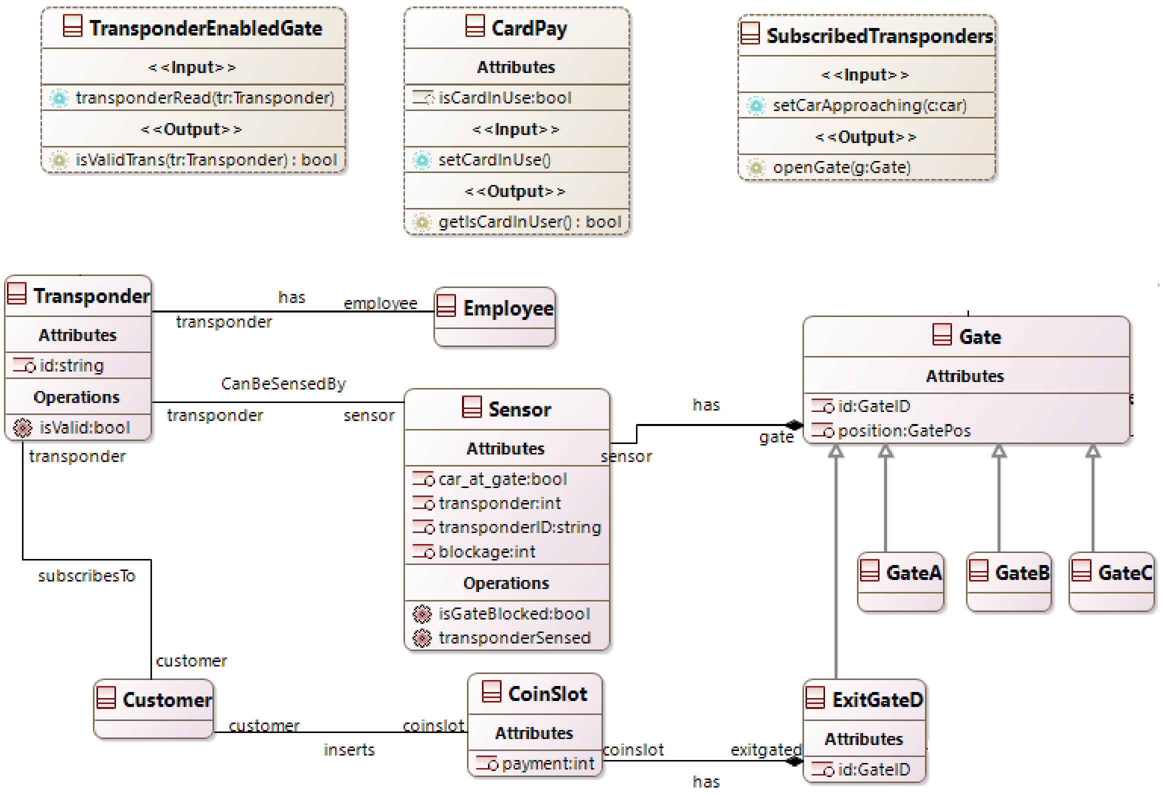


Figure 2.4: Partial Feature Class diagram for the GatedLot SPL.

2.2.3 Feature Module Diagram

For each feature, there is a Feature Module diagram (similar to the UML State-Machine diagram) that illustrates the behaviour of the feature [97]. The FORML Feature Module slightly adapts the UML State-Machine. The following describes only the important adaptations that are related to the context of our thesis:

- *World Change Event (WCE)*: refers to a traditional *event* defined in UML. There are three types of WCEs: 1) **C+(o)**: triggers the transition when the object *o* of type *C* is created (e.g., a message object has been created), 2) **C-(o)**: triggers the transition when the object *o* is removed from the world state (e.g., a message object is removed), and 3) **C.a(o)**: stimulates the transition when the value of the attribute *a* of the object *o* is changed [97].

- *World Change Action (WCA)*: is an action from the system that affects the environment [97]. The important types of actions discussed in this thesis are 1) $+\mathbf{C}(att_1 = exp_1 ... att_n = exp_n)$ that creates a C object (e.g., an output message object) with the given attributes att_i whose values are exp_i, and 2) **o.a := exp** that sets the value of the attribute a in the object o to value exp. Note that, a transition may have zero or more actions.

- *Fragment Feature Modules*: enable the evolution of existing Feature Modules. In many cases, a feature extends the behaviour of another feature by providing new capabilities [97]. In such a case, the two features share much of their behaviour, FORML allows the modeller to create a Fragment Feature Module for a new feature based on the Feature Module of an existing feature. For example, the *SubscribedTransponders* feature can extend the functionality of the *EmployeeTransponder* feature by performing an extra validation step to ensure that the transponder subscription is not expired. Specifically, the Feature Module of the *SubscribedTransponders* feature can be fragmented from *EmployeeTransponder*'s Feature Module, allowing the modeller to strengthen the guard of a particular transition to check the validity of a subscribed transponder. The goal of a Fragment Feature Module is to focus the modeller's attention only on the requirements being introduced by the new feature instead of modelling the whole Feature Module from scratch [97].

2.3 Modelling Frameworks and Tools

Fig. 2.5 illustrates an overview of the enabling technologies and model-editor frameworks that were used to develop our tool. This section presents the technologies: Eclipse Modelling Framework [11], Xtext [26], and Sirius [108].

2.3.1 Modelling Frameworks in Eclipse Platform

The foundational development environment that we used for our tool is Eclipse[1] as it has become one of the most widely used IDEs for the development of open-source integrated applications. Eclipse provides a number of frameworks for defining Domain Specific Modelling Languages (DSMLs) and developing graphical and textual editors for them. In the course of developing our tool, we used some of these frameworks such as the Eclipse Modelling Framework (EMF), the Graphical Editing Framework (GEF) [90], the Graphical

[1]see Eclipse website https://www.eclipse.org

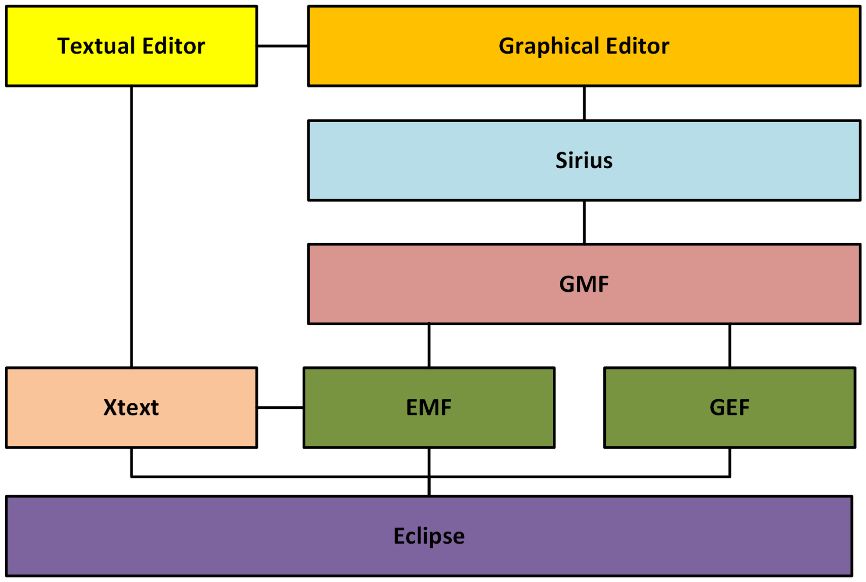


Figure 2.5: Overview of the enabling technologies stack and how they interact.

Modelling Framework (GMF), Xtext, and Sirius. The following are brief descriptions of these technologies.

- **Eclipse Modelling Framework:** The EMF[2] is the primary framework for Eclipse's modelling eco-system and is widely regarded as the standard for modelling in Eclipse [11]. In EMF, the developer can define a DSML's metamodel using a standard modelling notation, namely Ecore [11]. Furthermore, EMF provides facilitators for different purposes such as automated Java code generation based on the Ecore metamodel; and the EMF.Edit component, which is a basic editor for the defined language [11].

- **Eclipse Graphical Editing Framework:** A graphical editor visually presents the model elements and the relationships among them. The GEF provides the end-user with the drawing and graphical features such as: a tool palette, figures for the underlying model elements, the commands that a user can request (e.g., auto resize of a figure), and how the editor should respond to the commands (e.g., running a customized version of auto resize based on the selected figure in the diagram) [90]. We made little use of GEF as most of the functionalities are offered by the Sirius framework that is explained later in this section.

[2]see EMF website https://www.eclipse.org/modeling/emf/

- **Eclipse Graphical Modelling Framework:** The Eclipse community has proposed the GMF as their primary open-source framework for developing graphical editors for DSMLs [41]. GMF acts as a bridge between EMF and GEF. It listens to user's actions on the graphical editor and modifies the semantic EMF model accordingly (i.e., keeping the graphical and EMF models in sync). Moreover, it fetches the data from the EMF model and uses GEF to represent it in the graphical editor. GMF alleviates some of the efforts of developing a graphical editor imposed by GEF by abstracting the low-level implementation details in GEF.

2.3.2 Xtext: The Grammar Language

Xtext[3] is a widely used Eclipse framework for developing Domain-Specific Languages (DSLs) and editors. We used Xtext for developing the textual editor for our modelling language i.e., FORML. We fed the FORML grammar (see Appendix A) into Xtext and it generated the textual editor with several features such as syntax highlighting, (syntactical) error checking, and Content-Assist.

2.3.3 Sirius: A Tool to Create Graphical Modelling Workbenches

As mentioned, GMF is the main framework for developing graphical modelling editors in Eclipse. Although GMF reduces the efforts of designing graphical editors in GEF, it is still very complex and requires many low-level implementations. To overcome the challenges of working with GMF, we used the Sirius [108] framework instead. Sirius facilitates the development of graphical editors for DSMLs by abstracting the GEF and GMF details. It provides an easy-to-use user interface (e.g., wizards) which provides basic features to the user such as assigning figures to model elements, creating palette tools, setting drag and drop behaviours, defining right-click menus and the implementation for each menu item, and setting on-delete and on-create listeners. Although Sirius eases many of the challenges of using GEF and GMF, we were still engaged with the low-level implementations of GEF, GMF, and EMF for some of the advanced features in our tool. For instance, Sirius only supports the creation of basic figures (e.g., squares, rectangles, circles). Thus, when a user wants to assign a self-designed figure to a model element, she would still need to deal with the GMF implementation.

[3]see Xtext website https://eclipse.org/Xtext/

2.4 Cognitive Abilities and Theories

"Some may argue that HCI does not need theory. I disagree. Any discipline that fails to make a principled explanation to justify its practice is building on sand."

— – Alistair Sutcliffe, *"On the Effective Use and Reuse of HCI Knowledge"[106]*

Cognitive factors refer to one's characteristics that may improve or decline his/her performance and learning abilities (i.e., cognitive functions such as memory, decision-making, attention, and reasoning) [23]. This section briefly explains the most-relevant human-cognition theories that may affect the most-prominent challenges of modellers (as discussed in Chapter 4) and can be the basis of tooling enhancements (discussed in Chapter 5).

Different factors and challenges have been identified that need to be taken into account when designing software tools, especially in the field of Software Engineering (SE) [1], such as studying the impacts of social and organizational factors [38], analyzing users' mental models and how the tools address them [37], consideration of users' biases and preferences [85], and business considerations [20]. To address these challenges, researchers have started to employ a participatory design process [91], which takes into account all the stakeholders' requirements and analyzed their tasks and specific challenges from the very begging stages of the design [1]. Similarly, other techniques such as focus groups have been proposed that address the same concerns.

Furthermore, cognitive science has proposed different frameworks and standardized processes to address users' cognitive challenges. For example, Physics of Notations [69] proposes a set of techniques and theories to design and evaluate visual notations and representations. Similarly, Cognitive Dimensions Framework [37] proposes design and evaluation techniques and principles for visual coding environments.

However, tool vendors and researcher have barely employed cognitive theories to improve their modelling tools, partly because creating or enhancing a tool taking into account the aforementioned cognitive theories and knowledge is a significant challenge and is very costly [1]. In our work, we target several cognitive factors and theories that have been defined in [93] and [87], as explained below.

2.4.1 Memory

Users often face challenges when recalling contextual information from *memory* that is relevant to performing a modelling task (see Chapter 4). *Memory* refers to the ability of the brain to store and subsequently retrieve the information from our past experiences [93]. Memory storage and retrieval can be generally classified into the two main categories:

- **Short-term memory:** is a kind of memory storage that stores information for a short amount of time (usually less than a minute) [93]. For example, in our formative user study, we observed that the user often refers to the Class diagram, reads and stores some desired information (e.g., name of a class's attribute) in their memory, and then refers back to the State-Machine and retrieves and uses the recently stored information (e.g., using the attribute in a transition's guard expression). However, in many cases, we found that the subjects repeatedly stored (i.e., tried to memorize) some or similar information over and over to keep information in their short-term memory (referred to as *rehearsal* [93]).

- **Long-term memory:** refers to a type of memory storage that holds information for a longer amount of time (hours, days, months, or years) [93]. In modelling, it is possible that the modeller refers back to a partial model that she developed before to make it more complete. An example of using long-term memory in modelling domain is one of maintaining a model; that is, making extensions that fit with the rational of modeling that are captured in the current model. Therefore, the ability to recollect different pieces of information that were developed earlier can be very crucial in improving or declining modeller's performance in making new extensions.

Figure 2.6 shows how the information is maintained through the memory system. As shown in the figure, the first stage is to store the information into the short-term memory which can hold the information for less than a minute. After that, the information in the short-term storage are moved to the long-term storage though the *consolidation* process [93]. The consolidation process is usually boosted by recalling a memory over and over, thinking about the information, or sharing the memory with others [93].

2.4.2 Memory-Related Failures

Failing to recall information can decline the user's experience of performing a task, as revealed in our formative user study. According to Schacter and Gilbert [93], there are

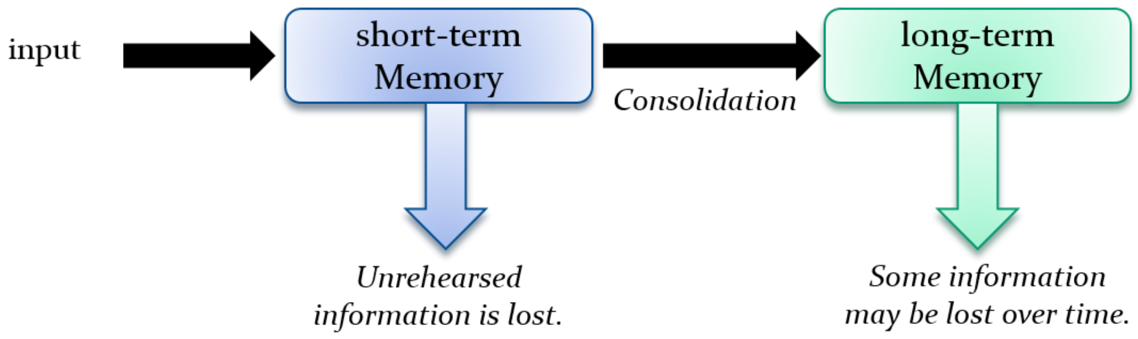

input
short-term Memory
Consolidation
long-term Memory
Unrehearsed information is lost.
Some information may be lost over time.

Figure 2.6: The maintenance flow of information in memory (taken from [93]).

several factors that can cause memory-related failures. The following explains a few of them that are relevant to our thesis context:

- **Transience**: refers to the disappearance of the stored information in memory with the passage of time [93]. That is, the quality of the stored information (e.g., details) generally deteriorates over time. Transience can happen because of interference [93]. In the context of modelling, it is often the case that the modeller edits different parts of the model and then needs to recollect the information related to each of these model segments at a later time. For example, a modeller may edit different classes in the Class diagram and then need to recall the details of these classes when she is editing the State-Machines for the classes. In such cases, it becomes more challenging (as time passes) for her to recall all the details of the model segments she previously edited.

- **Absentmindedness**: is related to not paying sufficient attention when performing a task [93]. In other words, a user *should remember to remember*. For example, a modeller should remember that using an undefined variable in a transition should be preceded by first defining the variable in the Class diagram, or it should be at least shortly followed by the variable definition.

- **Misattribution**: refers to the association of memories to the wrong sources [93]. In the context of modelling, it is possible that the modeller refers to a model segment that she thinks is part of the contextual information associated with her current editing task. However, the referred model segment may not belong to that contextual information. Thus, it may become frustrating to locate the model segment that contains the intended information. For example, a user may want to set the guard expression of a new transition to the guard expression of an existing transition from another State-Machine diagram. She may open the State-Machine that (she thinks) contains the intended information, but discovers that she has opened the wrong diagram. She may then need to search for the State-Machine diagram that contains the transition that she is looking for.

- **Lack of Context**: is one of the causes for the poor retrievals of some memories. It refers to insufficient amount of (relevant) contextual information or cues that are available to the person's focus of attention [17]. To understand context, we need to understand *Association*, which refers to the connection between two pieces of information [17]. For example, when editing a model element, it is important for the modeller to know about the model segments that will be impacted by an edit, so that she can check the impacted model segments against potential inconsistencies.

2.4.3 Human Errors

One of the major difficulties of modellers is to locate, understand, and resolve errors in models (see Chapter 4). To avoid or resolve an error, we first need to understand the types of human errors and their causes. According to the Generic Error-Modelling System (GEMS) [87], human errors can be categorised as *execution failures* and *planning failures* (see Fig. 2.7).

Execution failures occur when the user performs a task that deviates from the correct execution plan. Common types of execution failures are *slips* and *lapses*. Slips refer to commission errors where the user performs an incorrect or incomplete sequence of actions (usually due to inattention or confusion). Lapses refer to when the user omits one or more steps in a sequence of actions (usually because of memory failures). Planning failures occur when the user correctly performs the tasks but based on an incorrect plan. There are two causes of a planning failure: (1) application of an inappropriate rule (e.g., rules against the relevant specifications) or misapplication of a relevant/appropriate rule in the wrong situation (referred to as Rule-Based mistakes), or (2) planning based on uncertain or incomplete information (referred to as Knowledge-Based mistakes).

To clarify the aforementioned types of human errors, assume that a modeller uses an attribute in the State-Machine that is not defined in the Class diagram. According to GEMS, the four possible root causes of this error are:

- Lapse: if the modeller forgot to define the attribute in the Class diagram before using it.

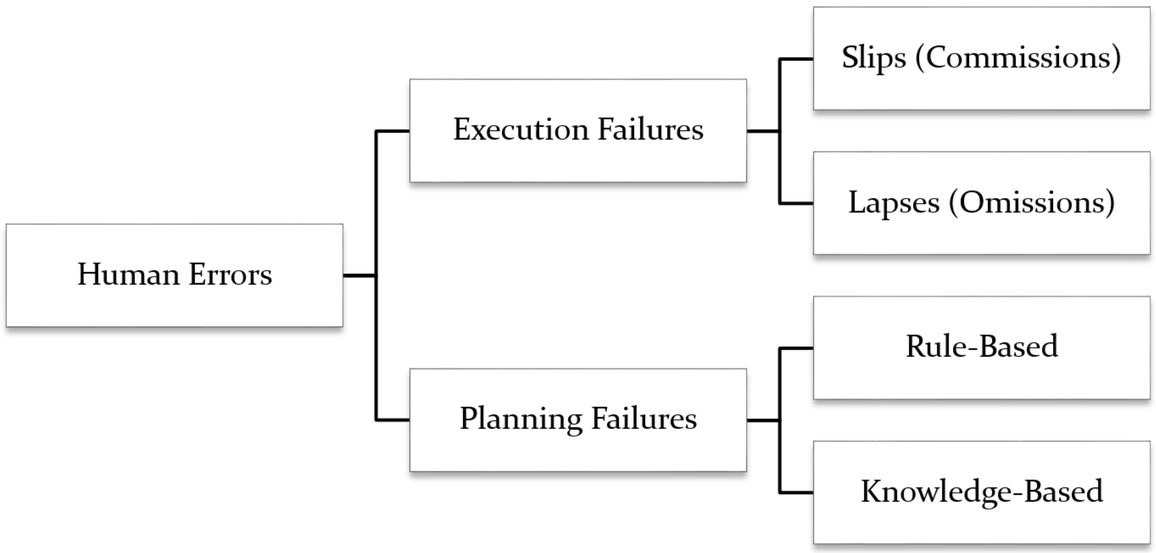


Figure 2.7: The categorization of human errors (taken from [87]).

- Slip: if the modeller knows she should first define the attribute but she did not care, or she misspelled an attribute name in a transition expression.
- Rule-Based: if the modeller preferred to use an undefined attribute at the moment and then define it at a later time.
- Knowledge-Based: if the modeller mistakenly thought that the attribute is already defined in the Class diagram.

2.4.4 Conclusion

In this section, we briefly described the important cognitive theories that can explain a user's cognitive difficulties that are relevant to our thesis. Moreover, knowing about the causes to memory-related failures and the types of human errors can help us invent tooling mechanisms that can improve a user's experience of editing and debugging models.

Chapter 3

Related Work

"User-Centered Design is a product development approach that focuses on end users. The philosophy is that the product should suit the user, rather than making the user suit the product. This is accomplished by employing techniques, processes, and methods throughout the product life cycle that focus on the user."

— – C. Courage & K. Baxter, *"Understanding Your Users"[10]*

This chapter explores the related work on tooling techniques that have been devised to reduce the efforts of modelling as well as improving the quality of modelling tools.

Investigating User eXperience (UX) in Software Engineering (SE) tools can be roughly divided into two categories: 1) UX in software development IDEs and 2) UX in software modelling IDEs. Although software-development communities appreciate the importance of UX and employ UX-related solutions as part of their IDE development (e.g., [2], [22], [30], and [49]), the software-modelling community is still in its infancy with respect to UX. Existing MDE tools offer User Interfaces (UIs) that aim to address issues related to usefulness of their editors, such as developing analysable models (e.g., [25], [34], [70], and [77]); but they lack a systematic consideration of UX metrics such as users' preferences and cognitive efforts.

Furthermore, current approaches do not empirically evaluate their techniques to assess their effectiveness (and usability) of their techniques on human users. Such ignorance of modellers and their tasks is one reason why basic general-purpose drawing tools such as Microsoft Visio or even Powerpoint are often preferred over the modelling tools for creating

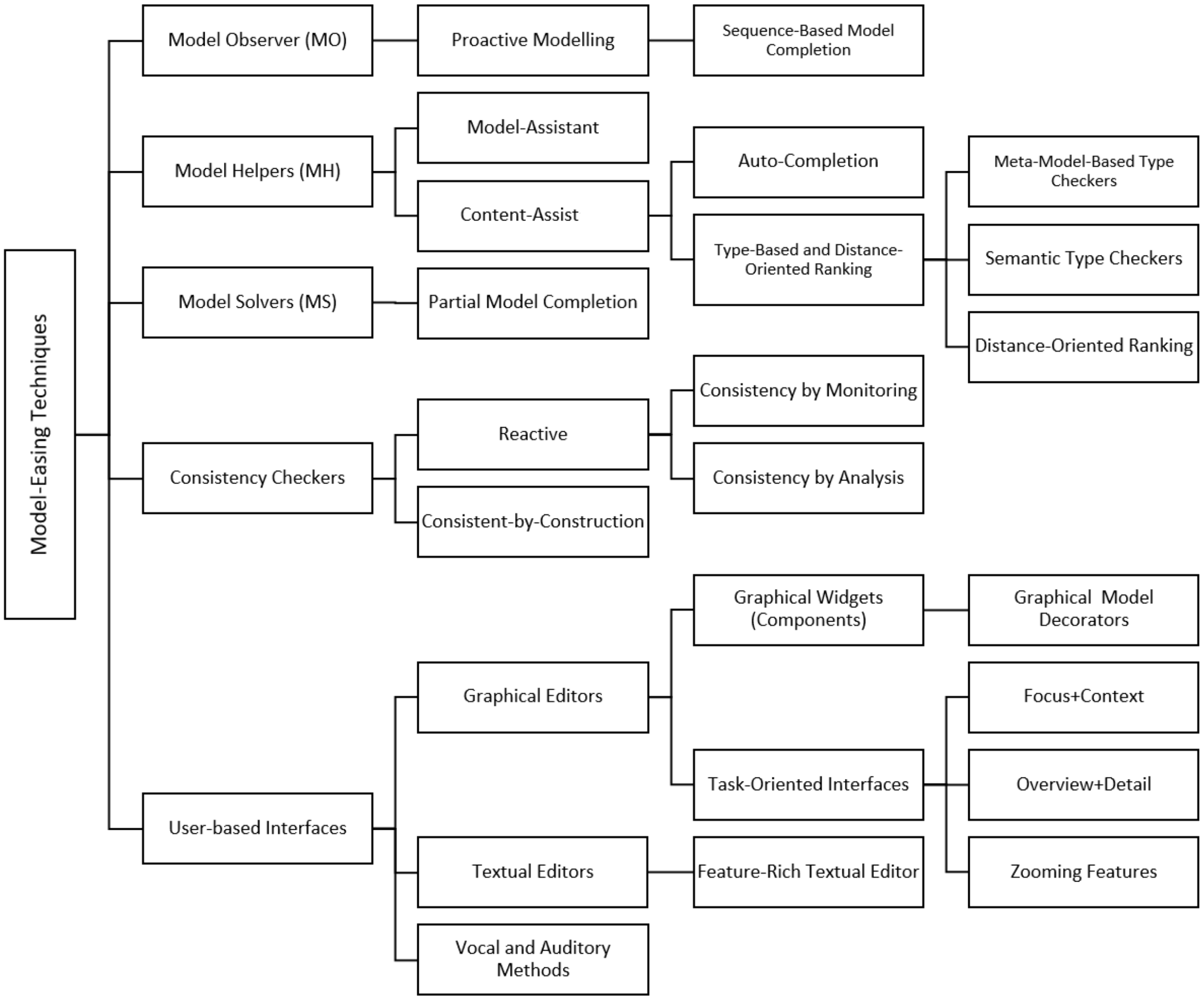


Figure 3.1: Feature tree for existing model-easing techniques.

graphical models [14]. According to Mussbacher et al. [71], users still experience strenuous efforts when using MDE tools, and many of them are reluctant to employ modelling mostly because they had a bad experience with a modelling tool.

According to Hill [43], the modelling effort can be defined as:

$$M = (Z + U + T) \tag{3.1}$$

where Z is the modeller's think time for a modelling task, U denotes the time that it takes a modeller to interact with the tool to realize the solution in their mind, and T denotes the time that it takes the tool to automate the requested task.

Furthermore, Hill [43] classifies the techniques that can reduce the modelling efforts into three categories: Model Observers, Model Solvers, and Model Decorators. We adapt and extend his classification into one with five categories: Model Observers, Model Helpers, Model Solvers, Consistency Checkers, and User-based Interfaces. Each of these techniques can be considered as *Artefact-Centric*, *User-Centric*, or both. Artefact-centric approaches mostly focus on reducing the user's think time (Z) by proposing techniques that take into account the model, the meta-model, and the properties of them. These techniques focus on details about the notations, syntax, semantics, and the model's conformance to these details. An artefact-centric technique does not *explicitly* aim at reducing the user's interaction time with the tool (U); however, it may *indirectly* impact the user's interaction time with the tool as well. A user-centric approach, on the other hand, takes an artefact-centric technique and makes it more human-friendly and usable. Unlike an artefact-centric approach that focuses *what to propose* to address the requirements specifications (e.g., UML specification), a user-centric approach focuses on *how to propose* a feature that optimizes user's experience of using the feature. Although, user-centric features manifest themselves in reducing users' interaction time with the tools (U), it is not the sole purpose of them. Moreover, the greater purpose of user-centric techniques is to emphasize on factors such as users' cognitive abilities, feelings, and quality of experiences that may be more important than users' interaction time.

In the rest of this chapter, we explain in more detail the classifications and existing techniques for reducing the effort of editing and debugging models and review the state-of-the-art for each of them. A feature tree of the techniques and their classifications is illustrated in Fig. 3.1.

3.1 Model Observers

A **Model Observer (MO)** continuously listens for model edits such as the creation, deletion and modification of model elements and then automatically reacts to the model edits according to the rules that have been specified in the language. The tool developer defines and embeds the follow-up actions that need to be performed automatically after a model edit. An MO can decrease a modeller's think time Z as he/she does not need to think about the actions that immediately follow a model edit.

Pati et al. [77] use MOs in their **Proactive Modelling** approach which proactively assists modellers by executing the valid next edits automatically. If there are multiple valid next edits, the tool prompts the modeller to select the appropriate one. For example, the tool developer can specify that, whenever the modeller creates a region in a State-Machine diagram, the tool should automatically create an initial pseudo-state for the region. They have embedded their MO techniques in the Generic Modelling Environment (GME) [58] as **add-ons**. Each add-on receives notifications of triggering edits and reacts according to predefined macros specified by the developer.

An MO can indirectly reduce the tool interaction time U as it automatically performs several tasks instead of waiting for the modeller to initiate every action manually. However, it is difficult to design such techniques as it is important to ensure that the completions are not disrupting the user's normal activities [77]. To address this issue, we provide a systematic approach for employing MOs in UML (and FORML), which interact with the user before applying the next valid steps automatically.

3.2 Model Helpers

A **Model Helper (MH)** provides advice to the modeller. For instance, a MH can recommend a list of model elements that can be used in a modelling task. Its primary goal is to reduce the modeller's think time (Z) by offering proposals of the *next* available edits. MH can be divided into Model-Assistants and Content-Assists, as explained below.

Model-Assistant is a type of MH that helps the modeller by offering model elements when needed. For example, Dyck et al. [25] propose a framework for command-enabled editors that explores online repositories, libraries, or knowledge bases (e.g., ReMoDD [32], MOOGLE [64]) to find some information, such as attributes of an entity, and then suggest them to the user. DoMoRe [5] is another similar approach that assists in developing Domain Models, which queries existing knowledge bases, ontologies, and natural language-based resources to retrieve information about a model element (using the element's name) and propose it to the user in a structured format (e.g., in terms of a possible association between classes). Although such MH techniques can reduce the modelling effort, it is overly simplistic when there are many model elements that can match the criteria, or no accurate information and resources are available for a domain. These types of approaches suffer from the cold start problem; that is, they can only reuse the available information that have already been gathered and converted into the repositories.

Another type of MH is known as Content-Assist that originated from programming IDEs. For instance, Content-Assist in Java [94] (or Intellisense in Visual Studio [84]) is a tooling feature that enables the developer to view a list of the possible next tokens that the developer is expected to write (at a specific cursor position) when writing Java code. The list is populated by the IDE and is based on the types of the elements declared in a program (e.g., variables or functions) or on the programming language's grammar. Model editors are starting to adopt such techniques and allow modellers to use Content-Assist when writing model expressions. Model editors such as Papyrus [34], Yakindu [70], and EMF enrich their textual editors by providing a Content-Assist feature, which suggests to the user a list of valid model elements (i.e., tokens or clauses) that can be referenced in a specific position when writing in the textual editor. The user can push the *Control* and *Space* keys in combination and the Content-Assist will provide a list of tokens that can be selected by the user to edit the model incrementally (token by token). A Content-Assist helps to reduce spelling errors and assists in recalling existing elements. Narrowing down and prioritizing the extensive list of elements (tokens) in a Content-Assist has been a research problem for several years [77]. To make the Content-Assists more clever, Bruch et al. [16] proposed a code-based approach that analyses the existing code bases and tries to predict the next command that the user may use. However, applying such approaches to modelling tools remains difficult because 1) there are not enough repositories for models to learn from, and 2) the level of details in modelling is a lot more abstract than coding, which makes the prediction of the next command an inherently difficult problem.

Existing Content-Assists for modelling tools (e.g., the one used in Capella [89]) mostly propose elements based on their type specified in the meta-model. For instance, the meta-model may specify that only class operations can be used in a triggering event. The Content-Assist will then only offer existing operations to the user when suggesting possible transition triggering events. Many IDEs such as Eclipse, however, refine their Content-Assist by enforcing semantic type soundness of elements when coding. For instance, if the Left-Hand-Side (LHS) is an assignment of type T, then the Right-Hand-Side (RHS) should also conform to type T. Similarly, existing Content-Assists place the elements of the current class at the top of the list of suggestions, to help the programmer see more quickly the elements that are more likely to be the intended element. The rationale behind this choice is that the modeller is more likely to use the elements of the current class than the elements of other classes.

In our thesis, we have devised an enhanced Content-Assist that filters out elements that do not make a semantically sound model and ranks the remaining ones based on their relevance to the element that is being edited. Although there are approaches that enhance Content-Assists in code editors, we are not aware of such approaches in model editors. Designing a Content-Assist for a model editor is more difficult than designing a Content-Assist for a code editor mostly because of the limited information that is available. For example, the absence of certain information such as types of elements in the model (due to the abstraction) makes it harder to define Content-Assist in model editors.

3.3 Model Solvers

A **Model Solver (MS)** is a formal methods-based technique that allows users to provide partial information (i.e., a partial model) and then invokes a solver, which tries to synthesize the rest of the model. Its primary goal is to reduce the modeller's think time (Z).

For instance, the partial model-completion techniques that are proposed in [95] and [105] determine potential valid completions of a partial (incomplete) model (e.g., State-Machine), based on a given meta-model and constraints on the model and the meta-model. Similarly, in Clafer [68], a user can provide a partial configuration of a Feature Model, and the tool provides the modeller with a list of all possible completions. However, a model-completion technique has the following limitations [77]. Firstly, the number of possible completed models can increase unmanageably for large models. Secondly, the inherent difficulties of developing models (e.g., using a tool to create a partial model in the first place) are not minimized.

3.4 Consistency Checkers

Software modellers face challenges to develop correct, consistent and complete models [39]. If errors go undetected and unfixed, it can lead to serious defects in a product. Model consistency-checking is a widely proposed solution and a number of approaches are discussed in the literature. The approaches can be divided into **Reactive** (or Corrective) versus **Consistent-by-Construction** (or Preventive) consistency-management techniques.

3.4.1 Reactive Consistency Management

Reactive Consistency Management refers to identifying and/or locating inconsistencies when the user demands or at different intervals. It is generally classified into two approaches [44]: 1) consistency by analysis, and 2) consistency by monitoring. In the following, we review major approaches that fall into this category.

1) Consistency by Analysis refers to techniques that detect inconsistencies through automated analysis of a model (mostly on user's demand). For example, in a model transformation-based technique [57], a statechart is transformed into a more formal model (e.g., a B formal specification) that is easier to check automatically. The consistency-checking is then performed by using tools (e.g., AtelierB, B-Toolkit) that can formally analyze the transformed model. Similarly, Xlinkit et al. [72] is a tool that manages inconsistencies among different documents that are developed in XML. The user can define constraints in first-order-logic and Xlinkit can check the model against the constraints.

As another form of consistency by analysis, Blanc et al. [12] propose **operation-based consistency-checking**, an approach for detecting model inconsistencies by defining Structural Consistency and Methodological Consistency rules. In Methodological Consistency, they record a modeller's actions, and then use predicate logic to find inconsistency problems by checking the prescribed order of actions against the order of the modeller's actual actions (e.g., create attribute and then set its name). Structural Consistency rules are used to find inconsistencies in the model itself, by looking at the structure of the model and detecting violations of rules that express desired structural properties.

The consistency by analysis approaches look for inconsistencies mostly at the user's request. Such approaches lead to extra effort to fix inconsistencies after the model has been developed [27].

2) Consistency by Monitoring refers to the techniques that locate the inconsistencies while modelling at regular intervals. One prominent type of consistency by monitoring is called immediate error detection, which refers to monitoring the model edits (using MO) and responding to the edits that introduce inconsistencies to the model. These techniques provide warnings to the modeller to make her aware of the existing errors. However, fixing the inconsistencies at the time of the detection is not necessary.

For example, Egyed [27] proposes a consistency checker, named UML/Analyzer, that detects model inconsistencies immediately as they are made. UML/Analyzer observes model changes (using MO) and checks them against the consistency rules that are relevant to the change. If any of the rules are violated, the UML/Analyzer warns the user with the inconsistency and its type. While the approach detects inconsistencies instantly, it does not require the modeller to fix errors right away. Therefore, this approach can be considered as a relaxed Consistent-by-Construction approach (i.e., instant detection but late resolution of errors), rather than a strict one.

The reactive consistency management approaches are promising, however they are not optimal because they allow the user to develop an inconsistent model. Thus, a modeller needs to expend extra effort to change the model and fix the inconsistencies at a later time [27]. Moreover, the fixes might themselves introduce new inconsistencies to the model, leading to additional delayed effort [27]. As we are interested in creating precise and analysable models, such reactive approaches do not fit our purpose. Instead, we use a Consistent-by-Construction approach (on-the-fly error resolution) to ensure that the created models are consistent at all times.

3.4.2 Consistent-by-Construction Consistency Management

Consistent-by-Construction modelling [102] either detects inconsistencies as they are committed and asks the user to fix them before proceeding or prevents inconsistency errors from being made in the first place. The key goal of Consistent-by-Construction modelling is to ensure that a model is consistent and correct at all times [44][102]. The superiority of this approach over the other consistency-checking approaches is that a modeller is not allowed to save an inconsistent model [102]. The hypothesis that underlies this approach is that correct-by-construction modelling mitigates the risk that inconsistencies can go unresolved and be carried through to the final product.

Ramesh et al. [86] propose the Extensible Real Time Software Design Inconsistency Checker (XRTSDIC) and argue that the XRTSDIC is capable of detecting model inconsistencies immediately based on a set of predefined consistency rules. If the modeller deletes an element from the Class diagram, the tool automatically deletes all the uses of the element. A similar approach is employed in most other proficient modelling tools such as Capella [89] and Papyrus [34]. Instead of automatically propagating the change to the model that may not be desirable to the modeller, we provide a user-interactive approach that distinguishes possible changes that can be made and reacts accordingly.

The aforementioned reactive and preventive consistency-management approaches provide sufficient solutions to the problem of detecting model inconsistencies. However, there are still many model editors that do not properly tackle the issue of resolving errors. Furthermore, some of the most advanced editors (e.g., MagicDraw [45]) provide a more relaxed environment that, in many cases, does not even detect inconsistencies in the model. Only a few Eclipse-based model editors (e.g., Capella [89]) address this issue by providing simplistic techniques that can detect errors immediately but they do not require the modeller to fix the inconsistencies immediately mostly because they are afraid to interrupt the user. On the contrary, we believe it takes less effort to fix the inconsistencies while they are being made than to detect and fix them at a later time, because the error-inducing part of the model is still fresh in the modeller's mind [27]. Therefore, we aim to support on-the-fly error resolution in order to improve the model's quality and reduce the efforts of consistency management. We note that, it is widely believed that maintaining inconsistencies at all times can reduce users' performance and become intrusive to the users [39][73]. We aim to mitigate the disruptive effects of on-the-fly error resolution by providing an interactive-interface that helps a modeller 1) to deal with inconsistencies without interrupting their work, 2) to collect information about the model elements that are affected, and 2) to avoid switching back and forth among the relevant diagrams to fix an inconsistency.

3.5 User-based Interfaces

3.5.1 Graphical Editors

Graphical Editors employ advanced visualization techniques that are related to the visual and spatial capabilities of a modeller, such as providing different graphical widgets and notations (e.g., icons) that can manifest a model element's internal state [43]. Graphical Editors play a significant role in decreasing the modelling efforts (using only the textual editors) [43] by reducing the modeller's think time as well as her interaction time with the tool (U).

3.5.1.1 Graphical Widgets

Graphical widgets refer to the graphical components that are used in Graphical Editors such as an icon, a button, a text-box, a table, or a graphical wizard. More specifically, a widget is a graphical element employed to construct a graphical UI or a view. For example, Eles and Lawford [28] propose representing variable expressions (like conditions or assignment expressions) in a tabular format that decomposes the expression into clauses (separated into distinct table cells), which are easier to write and modify. The decomposed clauses are easier to verify for correctness, completeness and disjointedness than it would be to verify the whole expression, as well as being easier to write.

Furthermore, Hill has shown that **graphical model decorators** can also reduce the effort of modelling [43]. A graphical model decorator is a visualization aspect of a model element that manifests the element's internal state [43]. Many domain-specific modelling tools [70][77] provide graphical icons and figures that better represent domain elements. For example, they represent a book entity in a library management system as a book figure. As another example, Yakindu [70] decorates defective model elements with an error-like icon to help the modeller understand the internal state of the erroneous model elements only by looking at the icons. However, the set of rules they embedded for detecting errors is limited and does not cover general and popular types of defects such as completeness rules.

Among industrial tools, ArgoUML and MagicDraw are two of the most popular ones. ArgoUML [88] is a UML graphical modelling tool that employs a combination of heuristics, user-centric features, and graphical user interfaces to provide to-do lists, non-modal wizards, search utilities, and tabular views. MagicDraw [45] is also a very powerful industrial tool that aims at model consistency-management and model-easing techniques such as handling cross-references among elements in a model, a well-designed UI, and support for collaborative model development. Similarly, Capella [89], Papyrus [34], and Yakindu [70] provide a well-designed modelling environment as well as some usability features and techniques to improve user-tool interactions, which make them user-centric. However, these tools do not make a concerted effort to aid all human-factor elements (e.g., user's working memory and learning style), which, if employed, would ease recollection of relevant (contextual) information, reduce context-switching among different diagrams of a model, and be more informative in the case of model-debugging and model-editing.

3.5.1.2 Task-Oriented Interfaces

We use our *memory* to recall the detailed information about past phenomena [93]. However, our memory can be overloaded when there are a lot of details [93]. For example, a modeller may not remember all the specific details and information that are relevant to developing a State-Machine. Most of the existing modelling environments provide users with features to improve understandability and navigability of the model's structure, but few features to aid with navigation of the semantics of (or the relations among) the artefacts [49]. As a result, modellers must rely on their memory to recollect the semantic information relevant to the fulfillment of a task.

A **Task-Oriented Interface** helps users deal with the cognitive load of recalling details by alleviating the user's task of navigating among software artefacts (called contexts) relevant to their present modelling task. Task-Oriented Interfaces can be categorized into three basic categories [21]: Overview+Detail, Zooming Features, and Focus+Context. These methods are explained below.

An **Overview+Detail** interface provides the user with two views of information, namely an overview and a detailed view [21]. Each of these views are displayed in separate display spaces [21]. Microsoft Word, PowerPoint, and Adobe Reader provide such an interface. For example, in PowerPoint, the user can view the overall presentation (sequence of slides) on the left side of the display and can click on any slide to view its detail in the main part of the application. The user can also change the space ratio of the presentation overview versus the slide detail in the middle of the screen. This interface style has become a standard in many applications including MDE tools (e.g., overview of a very large Class diagram as a side view). Moreover, the tools allow users to show or hide the overview, which suggests that designers believe that not all users may benefit from the overview [21]. As far as our thesis is concerned, such Overview+Detail interface barely reduces the effort of modelling as it does not help the modeller with the contextual information about the task.

A tool that employs **Zooming Features** typically does not display both overview and detailed information simultaneously. Rather, there is only one view and the user can zoom in and out to see specific details versus see an overview [21]. As an example, Glinz et al. [36] have proposed an integrated view for object-oriented modelling that focuses on *Abstract Objects* (instead of classes) and their respective State-Machines. An abstract object can be hierarchically viewed at different levels of abstraction such as views of collection of objects versus views of individual objects' behaviour. They provide zooming features that help users to view any aspect of the system in a greater or lesser detail. The user is allowed to navigate between two levels of hierarchy (structural and behavioural) by magnifying or demagnifying an abstract object. However, the shortcoming of zooming features is that the user cannot view the focus and the context(s) at the same time. Therefore, it still relies on the user's working memory.

An Overview+Detail interface views the same information separated in different views that correlate to different levels of abstraction. Similarly, a Zooming interface allows the user to switch among different abstractions of the same information by zooming in and out [21]. Both of these views still display the same information, whereas a **Focus+Context** interface allows the user to concurrently display and edit the focus and contextual information (i.e., different sets of information) in a single view [21]. To alleviate users' cognitive load, modellers can actively view related diagrams (i.e., context) to a particular task (i.e., focus), while working on the task itself. FlexView, proposed by Ghazi et al. [35], is an instance of the Focus+Context approach. Their method divides the screen into multiple regions, each of which presents a part of the information. This way, the user does not need to switch back and forth among different diagrams. As another example, Kagdi and Maletic [47] propose Onion Graphs to visualize large UML Class diagrams. The user can focus on a particular task (e.g., editing a class in the Class diagram) and view the context (e.g., other classes in the diagram) at different abstraction levels. Onion Graphs can be considered as a zooming feature for visualizing classes in a Class diagram; they have not been applied to other types of models such as State-Machine diagrams. Finally, in the Mylyn tool [49], Kersten proposes a **Task-Focused UI** that introduces an interface to allow a programmer to view the parts of the code that he/she previously edited for a particular programming task. Therefore, a programmer does not need to recall the details of her edits or look for the parts of the code and information related to the edits. Mylyn, however, focuses on code development rather than model development. We are interested in further investigating and adapting such ideas to model development.

3.5.2 Textual Editors

Textual Editors allow to view and edit a model in a textual manner. Moreover, it can be Feature-Rich and provide useful features to decrease the effort of writing complex models of formal languages. For example, a feature-rich textual editor can help a modeller develop a complex expression by providing syntax-checking, highlighting, and a Content-Assist. A modeller can select a model element in a graphical editor, and then view and modify its corresponding model text in a textual editor. The Content-Assist in a feature-rich textual editor provides valid proposals (e.g., tokens specified in the language's grammar) to the user when writing expressions. For instance, when writing an expression involving a navigation path, a Content-Assist ensures that the user navigates only to the next allowable options (i.e., elements) when searching for a desired model element. There are a few works that have implemented a textual editor in their tool [33][34][45][70]. Garzon et al. [33] propose a framework for model-driven engineering called Umple that reduces modelling effort by letting the user use either a textual or graphical editor, with edits in either view automatically propagated to the other view. However, their textual editor is not a feature-rich editor. RFSM [110] attempts to ease the development of regular expressions for software behaviours. It provides two features for users to construct expressions, namely a toolbar-based editor and a code-based editor. Similarly, Papyrus [34] and Yakindu [70] allow modellers to edit model elements using feature-rich textual editors combined with their graphical editor.

Most of the aforementioned tools are Eclipse-based, meaning that they are based on GEF and GMF. These tools use the correct-by-construction methodology to conform with the underlying GEF framework [14]; however, they do not address its largest disadvantage, which is the disruptive effects imposed on the users by enforcing the correct-by-construction methodology. That is, correct-by-construction is not desirable to some users as it may be perceived as being intrusive to them [77]. Therefore, it is important that the proposing editors are tested against their intrusiveness.

3.5.3 Vocal and Auditory Methods

Vocal and Auditory techniques allow users to perform modelling tasks by inputting voice commands to the tool. Also, the tool can give modelling feedback to the modeller via the system's sound speakers. This can reduce the time of the user's interaction time U with the tool. VoCoTo [53] and VoiceToModel [103] are examples of such features. This approach, however, may impose an overhead on the modeller to learn the underlying grammar and voice commands. A user study shows that, although the language can be designed to be simple and understandable to the users, some of the users find it difficult to remember the vocabulary [53].

3.6 Discussion

Some of the existing techniques discussed above such as Model Observers, Model Helper, Model Solvers, and Consistency Checkers alleviate the effort of editing and debugging models by focusing on reducing the user's think time (Z). Some other techniques go further and *reduce* (but not *minimize*) the user's interaction time with the tools (U) by proposing graphical and textual interfaces (i.e., they ease modelling through I/O). Apart from the Task-Oriented Interfaces, the aforementioned techniques do not take into account the users' behavioural profiles and human-cognition factors when designing their tooling features. Moreover, the majority of the existing techniques do not assess the effectiveness of their tooling features on human users. Thus, there might be a gap between what the users need and what the tools provide. This suggests research problems and opportunities in the improvement of modelling tools by taking human-cognitive factors into account. We believe that there are still opportunities to enhance the existing model-easing techniques, particularly MOs, MHs, Consistency Checkers, and Task-Oriented Interfaces. This enhancement can be done by taking two steps: 1) proposing enhancements based on the human-cognitive factors that influence modeller's cognitive efforts, and 2) conducting empirical studies to evaluate the effectiveness of the proposed techniques on human users.

Chapter 4

Understanding the Users

"Much of what makes a developer effective is the knowledge in their mind, but we know little about what this knowledge is, how developers acquire it, how to measure and model it, and how to use these models to improve tools or enable new categories of tools. There are many open opportunities in this space that could lead to powerful new understandings about software engineering expertise and powerful new tools to support software engineering."

— – Ko et al., *"Human-Centric Development of Software Tools"*[51]

Before proposing tool enhancements, it is important to understand what aspects of modelling and using model editors give users the most grief, and are most in need of enhancements. For this purpose, formative user studies [7] are devised in the literature, which aim at identifying the (negative or positive) issues with the tools by answering the following key questions:

- What are the most important reasons that prevent users to fulfill their tasks when using the product?
- What are the positive and negative aspects of the product?
- What are the challenges that users face when using the product?
- Can the product alleviate the users' challenges?

- What types of errors do the users make? Does the product help users with recovering from the errors?
- What is the users' level of satisfaction with using the product?

Accordingly, we conducted a formative user study to identify the most-severe challenges of users when using model editors. This chapter[4] presents the first step of our approach i.e., understanding the foremost difficulties experienced by model editor users. It includes the details of the context of our formative study and its design, as well as the observed results.

4.1 Experimental Context

According to the research, modelling tools are amongst the major barriers to the adoption of Model-Driven Engineering (MDE) in industry [46][71]. Hence, the main objectives of our formative study were to identify and learn about modellers' *difficulties* and *challenges* while performing modelling tasks and to provide an explanation for the challenges. Our focus was to learn about any problematic task in modelling software systems and the underlying causes of challenges. The result of the study can be used to direct our efforts in improving software modelling tools towards enhancements most likely to alleviate the difficulties of software modelling.

We conducted our formative user study on the UML model editors because the UML has become the prominent standard for modelling software systems [54]. The UML consists of several static and dynamic diagrams. It was not feasible to study all of the UML diagrams, so we focused the context of our study on a subset of the diagrams and selected two of the most-prominent UML diagrams: one static diagram and one dynamic diagram, namely the Class and State-Machine diagrams. We designed our user study towards observing the difficulties and challenges of modellers when developing the two mentioned diagrams. However, we believe that the results of the study may be applicable to similar diagrams of these types (e.g., Sequence Diagram or other variants of State-Machine models).

The goal of the study is summarized using the GQM template [107] (Table 4.1). The objective of the study was to assess the effectiveness of human users on using existing model editors. For this, we evaluated the tools with respect to five criteria: 1) *Effectiveness*: whether the tools enhance users' ability to successfully fulfill their modelling tasks, 2)

[4]This chapter is reprinted in adopted form from [80].

Analyze	the tasks of Class diagram and state diagram modelling
For the purpose of	understanding and learning about fundamental difficulties and challenges
With respect to	the effort of modelling
From the perspective of	software modellers
In the context of	existing technologies and tools for UML Class and State-Machine diagrams modelling.

Table 4.1: Goal of the study according to GQM template.

Efficiency: whether the tools help in reducing modellers' effort (e.g., time) to perform modelling tasks, 3) *Satisfiability of Users' Expectations*: how big the gap between users' expectations of model editors and the model editors' capabilities is, 4) *Users' Satisfaction*: the extent to which the users are satisfied with using the tools, and 5) *Users' Challenges*: the most-severe challenges experienced by users when using model editors. To address the objectives of the study, 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: What are the most-severe challenges experienced by modellers employing modelling tools?*

The above 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, 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.

4.1.1 Pre-Study Phase

In the pre-study phase, we examined 30 models that had been submitted as solutions to a modelling assignment (see Appendix B) 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. Students 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 models. We used this as a guideline to estimate how much of the modellers' difficulties were actually due to expressing the domain/system description in the modelling notation.

2. *Model Quality* – detecting errors in the model. We grouped model-quality errors into different categories, listed in Table 4.2. Category 1 errors refer to structural inconsistencies (e.g., a Region in a State-Machine without any initial pseudo-state), whereas Category 2 errors refer to semantic-related errors (e.g., using an element without defining it). Category 3 errors refer to syntactical inconsistencies, and Category 4 errors refer to the type-soundness of a model (e.g., type-mismatch in the guard expression of a transition).

4.1.1.1 Pre-Study Results

Our 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 described problem as a basic UML State-Machine that captured the essence of all the described behaviour.

However, our evaluation of the models' quality suggested that students had difficulty creating correct and consistent models at the expected level of detail and precision. Table 4.2 presents the number of subjects that committed errors of each error type, and how

Table 4.2: Summary of the results of the Pre-Study analysis.

Category ID	Description	No. of Errors	No. of Subjects
Category 1	Incorrect use of the structure of UML models (e.g., a State-Machine without an initial pseudo-state).	7	4
Category 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
Category 3	Wrong or inconsistent use of UML notation and syntax.	27	8
Category 4	Type mismatch between the left-hand-side (LHS) and right-hand-side (RHS) of an assignment or condition.	42	9

many instances of each error type were made in the 30 models. In some cases, such as Category 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 no such element existed in the Class diagram.

Table 4.3: 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 and RHS expressions 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 4.3). Because students made few Category 1 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 in the pre-study's outcome of the potential critical challenges. In addition to these challenges, Reason [87] 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 from the pseudo-state to the initial state, etc.). Accordingly, we include *Order* in our list of challenges. We also added *Debugging* as a significant challenge because we observed that some errors could be avoided if a student invoked a tool's debugging feature. Subsequently, we scoped the tasks in our main study to focus on these suspected challenges.

4.2 Experimental Design

To investigate the aforementioned research questions, we conducted a formative user study to understand which aspects of performing modelling tasks using a model editor give modellers the most trouble. In the rest of this section, we explain the details of our experimental design. The design of our experiments is aligned with the guidelines from different sources [7],[52],[99].

4.2.1 Population and Sampling

The target population of the experiment was software modellers who develop models of UML-like languages. However, it is always difficult to recruit industry-level modellers. We used a mixture of *convenience sampling* and *purposive sampling* approaches to recruit participants [7]. Convenience sampling means that we recruit participants from a convenient subset of the larger population, whereas purposive sampling narrows down the recruited participants based on predefined criteria to ensure that the capabilities and knowledge of the sample reflects the larger population. Accordingly, we recruited students who have the required knowledge and experience. After almost six months of recruiting, we enlisted 18 subjects which has been shown to be adequate [7],[52],[99].

4.2.2 Recruitment Procedure

We emailed a recruitment letter[5] 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

[5]Please refer to Appendix C to see all the materials used for the formative user study.

Software Engineering, Computer Science, and Electrical and Computer Engineering at the University of Waterloo; we considered these students a good fit 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.

4.2.3 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 [63] their familiarity with UML Class and State-Machine diagrams as well as their competency and experience with the modelling tools.

In addition, the recruitment letter included 10 UML-specific questions selected and adapted 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.

An overview of the subjects' occupation, UML familiarity, and UML tool experience is provided in Table 4.4. It shows that all of our subjects, except one, self-expressed adequate familiarity with the UML. The table also shows that all of them had adequate experience with using UML modelling tools. Therefore, the selected subjects could be considered as a fair representation of the larger target population of the study.

4.2.4 Compensation

In appreciation of the subjects' time and commitment, they received an honorarium of $20 and were given the chance to enter into a draw and win a prize of $200 gift card.

4.2.5 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

Table 4.4: Subjects' demographics, backgrounds, and UML familiarity.

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
Class Diagram Familiarity	Fairly Familiar (Novice)	1
	Familiar	3
	Very Familiar	10
	Strongly Familiar (Experienced)	4
State-Machine Diagram Familiarity	Fairly Familiar (Novice)	1
	Familiar	9
	Very Familiar	5
	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

to study a textual description of the domain as well as the Class diagram of the system before starting the study's tasks. The subjects were allowed to refer back to the description during the experiment whenever they needed.

4.2.6 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 (e.g., MagicDraw, ArgoUML, Astah, Visual Paradigm, UMLet, Umple). Moreover, we excluded those subjects who chose to work with a tool that was more of a drawing tool than a modelling tool (e.g., Microsoft Visio). Figure 4.1 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 [4].

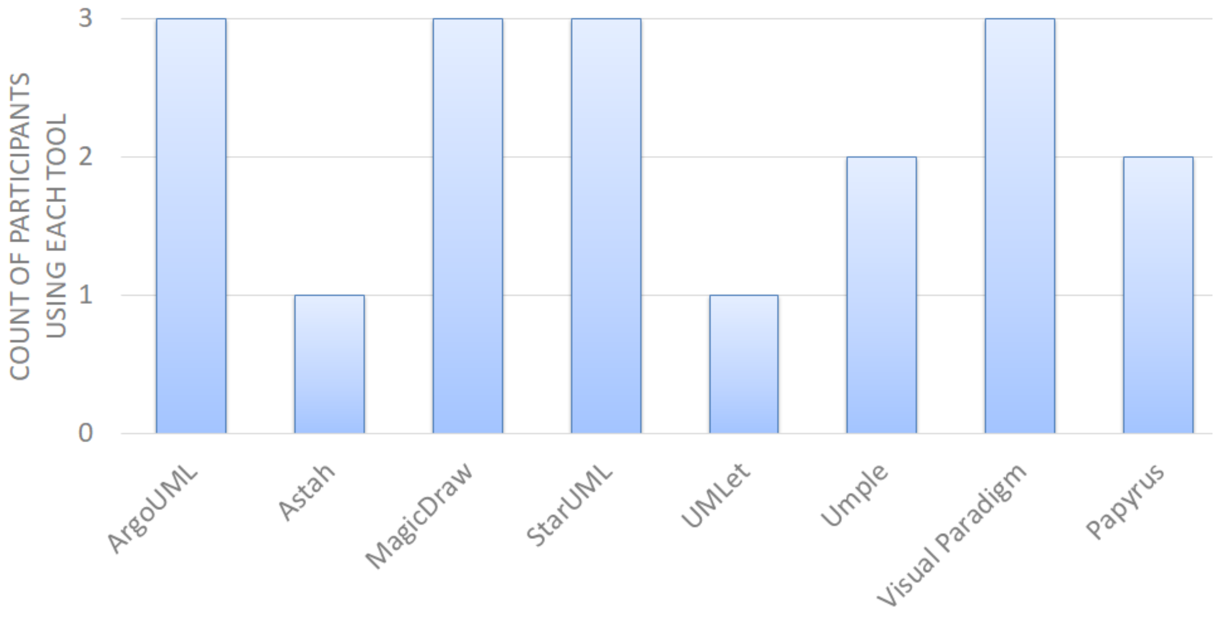


Figure 4.1: Number of subjects per tool.

4.2.7 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 (see Appendix C). 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., *Task 1, Task 2, Task 3, and Task 4*), 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.

Task 1: 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.*

The above model-editing task asks the subject to set the guard and action segments of the transition *T1.* The main goal of this task is to evaluate the level of difficulty that the subject faces with respect to recollecting the Class diagram's elements (e.g., *id* attribute of the *gate* class), and gauge the amount of help that is provided by the tool to the subject to implement the solution. For instance, whether the tool can help the modeller avoid spelling mistakes (which are trivial to catch and report) when writing the elements' names or not.

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, *Task 5* asks the subjects to rename elements in the Class diagram, and then locate any inconsistencies in the model that are introduced by that action. *Task 6* and *Task 7* ask the subjects to locate inconsistency errors embedded in the model, such as model elements that were used but not defined, and to detect incorrect navigation expressions. Following is a sample model-debugging task (i.e., *Task 5*).

Task 5: 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 that are caused by this change.

4.2.8 Data-Collection Techniques and Design

We evaluated the subjects' performance along three different dimensions, as prescribed by Tullis and Albert [7], namely performance, self-reported, and behavioural metrics. This section precisely defines the variables and metrics measured in the study.

4.2.8.1 Performance Metrics

We measured three performance metrics as described below.

- **Task Completeness (Success score):** This is a universal metric that shows how effective a subject was in completing a task [7]. It provides compelling evidence that there is something wrong if a subject cannot successfully finish their task. We used Level of Success (o, 0.5, 1) [7] to measure 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 are performed (i.e., the requirement of a task is correctly addressed by the solution). 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 completed a task but received help during the task, such as assistance with the language syntax or clarification about a model element in the application domain.
    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*: This is a traditional metric in the usability literature [7] that reports the mean average of the success scores for all of the subjects. More specifically, it refers to the summation of the subjects' score of completion 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). For instance, if 15 out of 20 subjects successfully completed the *Task 1* and 5 other subjects received help during the task (i.e., partially completed), then the *generally complete* score for the task is calculated as $(15 * 1.0) + (5 * 0.5) = 17.5$.

- **Errors:** We measured the number of errors as well. We classified the errors based on a taxonomy of Well-formedness and Consistency types of errors presented by Lange *et al.* [54]:

    - *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.

For each subject, we counted the number of errors of each error type. Note that, those errors where the subject failed to correctly model the requirement of a task are considered as incomplete task.

- **Efficiency:** We also measured the level of efficiency of the subjects per task. In our study, we used Lostness and Time-on-Task metrics, which are described below.

1. *Lostness* measures how *"lost"* a subject is when performing a task. The following three factors contribute to an assessment of lostness: 1) R: the minimum number of diagrams or dialogues that must be visited to accomplish a task, 2) S: the total number of diagrams or dialogue-boxes visited while performing a task (counting revisits), and 3) N: the number of different (unique) diagrams or dialogue-boxes the subject visited while performing a task. Lostness, L, is then calculated using the following formula [7][101].

$$L = \sqrt{(\frac{N}{S} - 1)^2 + (\frac{R}{N} - 1)^2} \tag{4.1}$$

Lostness scores range from *Zero* to *One*. The higher the score, the more trouble the user had finding what they want. Smith [101] 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 relative lostness by comparing a subject's lostness score to the optimal score (e.g., to the smallest value of lostness among all the subjects) [7].

2. *Time-on-Task* is the most popular way of measuring efficiency. It refers to the time that it takes a user to perform a particular task using a product. In addition, it can also be thought of as the modelling effort based on Hill's definition of the modelling effort [43] shown in Chapter 2.

4.2.8.2 Self-Reported Metrics

Perhaps the most obvious way to learn about the usability of a tool is to ask users to tell us about their expectation and experience with the tool [7]. We designed our experiment to collect self-reported data from a user for gauging their level of satisfaction.

Expectation versus Experience ratings: Albert and Dixon [6] propose an approach to assess users' reactions after a task. More specifically, they argue that it is important to collect data about how easy or difficult a task was compared to how easy or difficult the user thought or expected it to be. Accordingly, we asked the subjects to provide two ratings (using a 7-point rating scale of 1=Very easy to 7=Very difficult for both ratings).

1. their expectation rating; that is, their 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 after each task; that is, their rating of how easy or difficult the task actually was.

For each task, we then calculate an average *expectation rating* and an average *experience rating* of all the subjects. Comparing the two average ratings, we can then understand how effective the tools are in satisfying the users' expectations and needs (see Fig. 4.2 for a depiction of the possible outcomes of such a comparison). For instance, if the user expects a feature to be easy but he experienced it to be very difficult, then the feature is not addressing users' expectations and needs; thus, the improvement related to the feature should be addressed immediately and marked as "fix it fast".

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

- *Pre-Session Expectation Rating:* As mentioned in Section 4.1.1, we created a preliminary list of challenges that a user might face when using a tool. Before starting the session, we collected data about each subject's overall expectation of a tool's proficiency with respect to each pre-study challenge.

- *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 use the averages of the pre- and post-session ratings to get insight into opportunities for improving the tools.

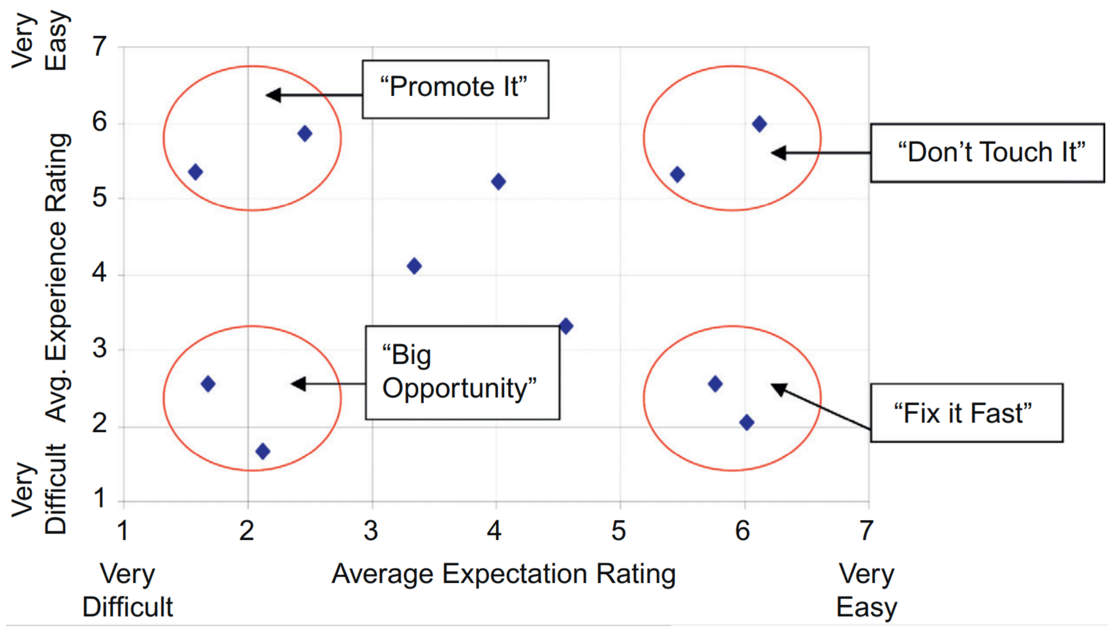


Figure 4.2: Comparison of expectation vs. experience ratings (taken from [7]).

- *Pre-Task Expectation Rating:* Before the start of all tasks, we asked the subject 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 the subject to rate how easy or difficult the task was based on their actual experience of using the tool. The main goal of the pre- and post-task ratings is to provide insight into the gap between a subject's expectation and what the tools actually provide.

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) [15], Computer System Usability Questionnaire (CSUQ) [61], Questionnaire for User Interface Satisfaction (QUIS) [19], and, Usefulness, Satisfaction, and Ease of Use Questionnaire (USE) [65]. Among these, we decided to use CSUQ [60][61] because the questions listed in the CSUQ were more in line with the goals of our study and needed less adaptation than the questions in the other questionnaire types. 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 their agreement on a 7-point scale of "Strongly Disagree" to "Strongly Agree", plus N/A. We allowed the user to select N/A wherever they felt unsure about their answer because not all of the questions were applicable to all of the tools.

4.2.8.3 Behavioural Metrics

In our study, we used the think-aloud protocol which is the best way to facilitate identifying usability issues during an in-person study [10]. We asked the subjects to express their thoughts loudly as soon as they occur when performing their tasks, so that we could understand their cognitive difficulties. Moreover, for the purpose of later analysis, we decided to screen-capture their tool use during the session as well as audio-record their voice.

The think-aloud protocol allowed us to collect precious information from the subjects' verbal expressions 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.
- Subjects ***not*** saying or doing something that they should have done or said.

To get the most out of the subjects, we prompted the subject if they did not express their 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?* Also, to ensure that our timing data would be accurate, we tried to avoid having a discussion during the performance of tasks, but in between tasks.

The recorded voice of the subject along with recorded video of the screen later helped us to extract the behavioural metrics for the experiment. After each session, we analysed the audio recording of the subject's session to identify the most valuable verbal expressions, and expanded the list of the significant verbal expressions from all subjects. 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.

4.3 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 and ends when the subject acknowledges that 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 five main segments: 1) Preparing the subject, 2) Collecting the Expectation Ratings, 3) Performing the Tasks, 4) Collecting the Experience Ratings, and 5) CSUQ. We describe these segments in more detail as follows. The design of the experiment is shown in Fig. 6.3.

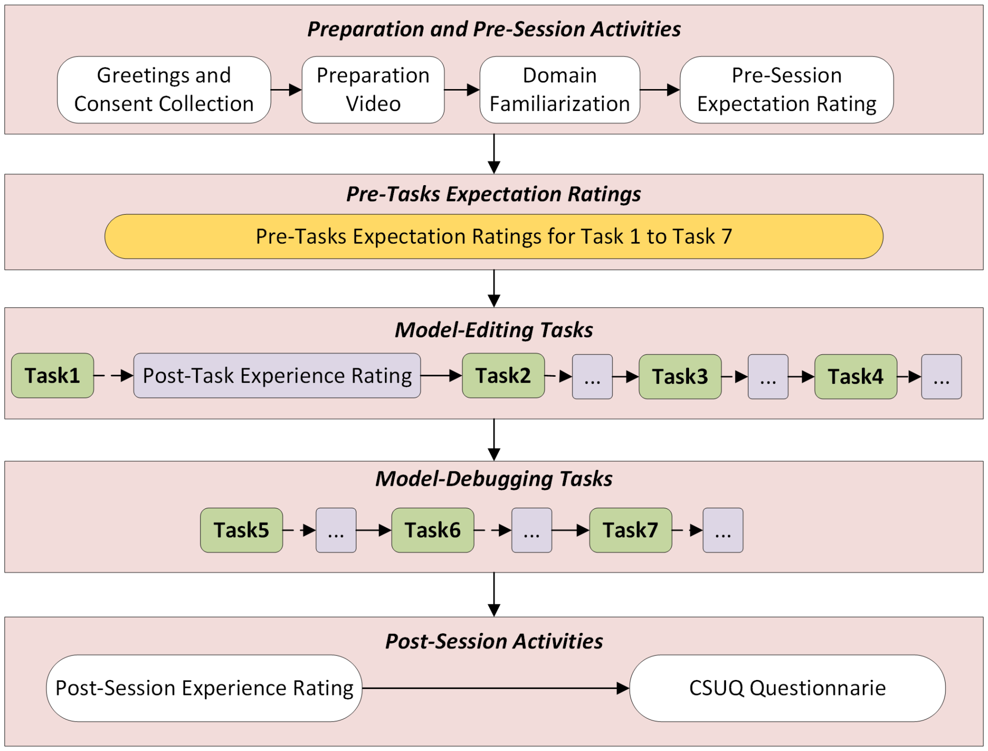

Preparation and Pre-Session Activities
Greetings and Consent Collection
Preparation Video
Domain Familiarization
Pre-Session Expectation Rating
Pre-Tasks Expectation Ratings
Pre-Tasks Expectation Ratings for Task 1 to Task 7
Model-Editing Tasks
Task1
Post-Task Experience Rating
Task2
...
Task3
...
Task4
...
Model-Debugging Tasks
Task5
...
Task6
...
Task7
...
Post-Session Activities
Post-Session Experience Rating
CSUQ Questionnarie

Figure 4.3: Experimental design.

4.3.1 Preparing the subject

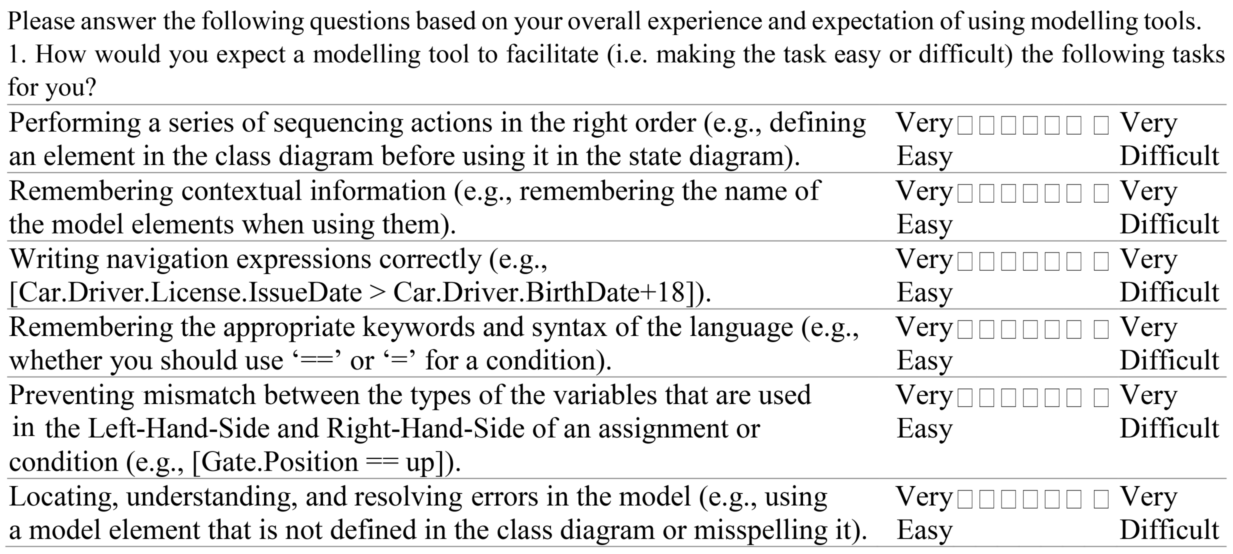
After greetings and signing the consent form, each subject was given an introduction to the experiment by viewing a preparation video. The subjects were asked to watch and listen to the video carefully. The video included general information about the procedure and methods of the study (e.g. think-out-load method). Appendix C contains the script of the preparation video which was prepared based on the guidelines from [10]. The advantage of using one video for all the subjects was that they all received the same information with respect to getting prepared for the study. Moreover, in the video we emphasized that subjects, in the course of the experiment, would not be evaluated in anyway; that it was the modelling tools that were under scrutiny and not them. We needed the subjects to understand this because it was important to create a relaxing and informal atmosphere to make the session as effective as possible.

Once the general information about the study were given to the subjects, we asked them to study the domain description of the Parking Lot system. The domain description was important because the subjects needed to become familiar with the system and understand the Class diagram and the model elements (e.g., classes, operations, attributes) in the Class diagram. We also provided them with descriptions of all the model elements to give the subjects a good understanding of the Parking Lot system, the Class diagram and the entities inside it (See Appendix C). There was no time constraint on the subjects for reading the description.

4.3.2 Collecting the Expectation Ratings

Before the subjects started the model-editing and model-debugging tasks, we collected some data on pre-session expectation ratings and pre-tasks expectation ratings. In the pre-session expectation ratings we asked the subject to rate their expectation of how difficult a particular modelling challenge (identified in the pre-study described in 4.1.1) would be. The list of the questions for the pre-session expectation ratings is shown in Fig. 4.4.

We also asked the subjects to read the description of each task and rate how easy or difficult they thought the task would be. We did all the pre-task ratings at the beginning of the session because we wanted to avoid the learning effect that performing a task could have on the ratings of the next task.



Please answer 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 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 mismatch between the types of the variables that are used in the Left-Hand-Side and Right-Hand-Side of an assignment or condition (e.g., [Gate.Position == up]).	Very Easy ☐☐☐☐☐☐ ☐ Very Difficult
Locating, understanding, and resolving 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 4.4: Pre-session expectation rating questionnaire.

4.3.3 Performing the Tasks

After collecting the information on the expectation ratings, we asked the subject to perform the tasks in the modelling tool that was open and prepared for them to work. Please remember that each subject only used the tool that they chose for this study.

As mentioned, we used the think-aloud protocol to identify the subjects' behavioural challenges. We used *Snagit* software to screen-capture the sessions and used a recorder application to record their voice. While some of the subjects were expressive enough, some others had difficultly thinking aloud while working with the tool. In such cases, we tried to motivate them to talk by asking questions regarding what is currently happening in their mind.

We also found that there is a trade-off between keeping the quality of the task solutions high and imposing a strict time constraint on a task. We tried not to impose any time pressure on the tasks in order to keep the quality of the task solutions high. Rather, we allowed subjects to announce when they completed a task. Although this could help in coming up with a satisfactory level of the quality of the tasks' solutions, the timing results could become unrealistic and useless for analysing times on tasks. To overcome this challenge we took the following strategies:

- If the subject insisted on continuing a task task but was not showing any signs of progress, then the researcher stepped in and provided some hints to the subject. If the subject subsequently completed the task, the task success would be 0.5. If after giving the hints, the subject could not fulfil the task, then we asked them to move on to the next task and the task success would be 0.

- We did not offer an hourly rate. Our offer was a fixed honorarium of $20 based on an estimate of a maximum of 90 minutes of work for the study. Hence, the subjects knew that they will get the same amount even if they fulfil their tasks sooner than expected. This strategy could encourage the subjects to finish their tasks as quickly as possible. It also avoided any incentive to play around with the tool to receive a higher payment.

- The subjects could leave sooner if they finished their tasks. Therefore, the subjects did have some motivation to finish their tasks as quickly as possible.

The subjects were asked to answer the post-task experience ratings for a task when they were done with it. The post-task experience ratings were designed to understand the subjects' view of how easy or difficult a task was based on their actual experience.

4.3.4 Collecting the Experience Ratings and CSUQ

At the end of the session, we asked the subjects to respond to two questionnaires: a post-session experience rating and a CSUQ. The post-session experience ratings consisted of the same set of the questions that were shown in 4.4, but this time asking the questions based on the subjects' actual experience of using their chosen tool. The CSUQ collected useful data about each subject's level of satisfaction of using their chosen tool.

In addition, at the end of each session, we asked the subject to provide any additional comment on the tool and their experience of using the tool. This could be any sort of positive or negative feedback.

4.4 Results

This section presents the results of the study in terms of each metric.

4.4.1 Tools' Effectiveness (RQ.1)

RQ.1: How effective are tools in communicating with users to improve the experience of performing modelling tasks and the correctness of models?

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

Task Success: The results of the subjects' success in completing the tasks are presented in Table 4.5. A cell value of 1 indicates that a subject had full success on a task; a cell value of 0.5 indicates that a subject completed a task with assistance; and a cell value of 0 indicates that a subject failed to complete a task. Fig. 4.5 illustrates the subjects' success rates on the tasks and shows that a significant number of subjects were not successful in completing their tasks without assistance. More importantly, it shows that fewer than 45 percent of the subjects successfully completed tasks 6 and 7, which were related to the debugging of models. This suggests that the tools do not provide enough support to help subjects resolve errors in the models.

Errors on Tasks: Table 4.6 lists the number of errors each subject made over the course of the study. Different errors were made by different subjects but some of the common errors were: referring to an undefined element in a State-Machine diagram, misspelling the name of an element when referring to it, wrong navigation paths (e.g., g.blockage instead of g.Sensor.blockage), wrong guard syntax (e.g., using == instead of =). We found that most of these error were due to the fact that the tool did not actively provide consistency checking for the user and the user relied too much on the tool, assuming that the tool would issue warnings.

As can be seen in the table, none of the subjects finished all of their tasks without error (see also Fig. 4.6). More importantly, the error rate shows that almost 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 relatively salient taking into account the fact that the tasks were generally easy and the model was very small. One can extrapolate the result to a complex model and notice how severe it will be.

4.4.2 Efficiency (RQ.2)

RQ.2: How efficient are modellers when using modelling tools?

Efficiency was investigated by means of two metrics: time on task and lostness.

Table 4.5: Results of each subject's success in completing each task.

Participant	Task1	Task2	Task3	Task4	Task5	Task6	Task7
P1	0.5	0.5	1.0	1.0	1.0	0.0	1.0
P2	0.5	0.5	0.5	1.0	0.5	0.0	0.0
P3	1.0	1.0	1.0	0.5	1.0	0.0	1.0
P4	0.5	1.0	1.0	0.5	1.0	1.0	0.0
P5	0.5	0.5	0.5	0.5	1.0	1.0	0.0
P6	0.5	1.0	1.0	1.0	0.5	0.0	0.0
P7	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P8	1.0	1.0	1.0	1.0	1.0	0.0	0.0
P9	0.5	0.5	0.5	0.5	0.5	0.0	1.0
P10	0.5	1.0	1.0	1.0	0.5	1.0	1.0
P11	1.0	0.5	1.0	1.0	0.5	1.0	1.0
P12	1.0	1.0	1.0	1.0	1.0	0.0	0.0
P13	1.0	1.0	1.0	1.0	0.5	0.0	0.0
P14	1.0	1.0	1.0	1.0	1.0	0.5	1.0
P15	1.0	1.0	1.0	0.5	1.0	1.0	0.0
P16	1.0	1.0	1.0	1.0	1.0	0.0	0.0
P17	1.0	1.0	1.0	1.0	1.0	0.0	0.0
P18	1.0	0.5	0.5	1.0	0.0	0.0	1.0
mean	0.81	0.83	0.89	0.86	0.78	0.36	0.44
median	1.00	1.00	1.00	1.00	1.00	0.00	0.00
min	0.50	0.50	0.50	0.50	0.00	0.00	0.00
max	1.00	1.00	1.00	1.00	1.00	1.00	1.00
stdev	0.25	0.24	0.21	0.23	0.31	0.48	0.51
Full Success Rate	61%	67%	78%	72%	61%	33%	44%
Partial Success Rate	39%	33%	22%	28%	33%	6%	0%
General Success Rate	81%	83%	89%	86%	78%	36%	44%
Failure Rate	0%	0%	0%	0%	6%	61%	56%

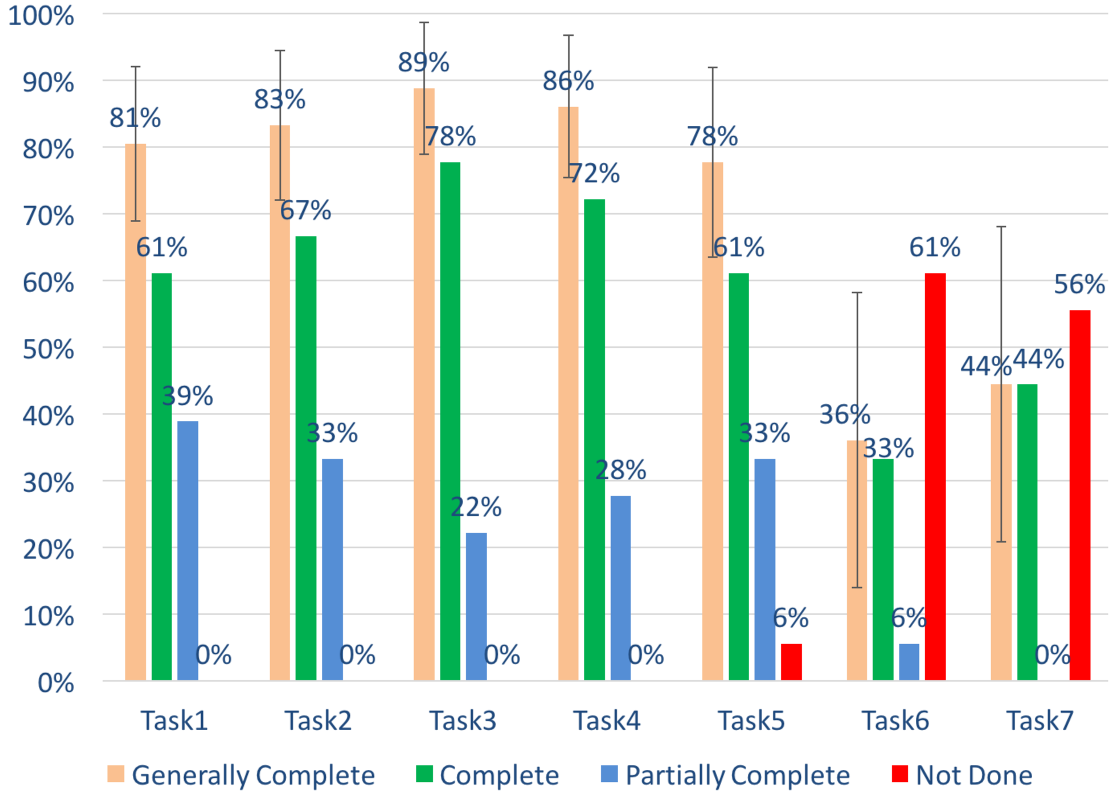


Figure 4.5: Task completion rate (error bars represent 95% confidence interval).

Table 4.6: Result of the subjects' errors on each task.

	Participant	Task1	Task2	Task3	Task4	Task5	Task6	Task7
Wellformedness	P1			1	1			
	P2				1			
	P3		1	1				
	P4			1				
	P5	1	1	1				
	P6		1	1				
	P7							
	P8	2						
	P9							
	P10	1	1					
	P11							
	P12	1	1	1				
	P13	1						
	P14	1						
	P15							
	P16							
	P17							
	P18	1						
Consistency	P1	2	2	2			1	
	P2	1					1	1
	P3	3	1	1	1		1	
	P4	1			1			1
	P5	1	1	2	1			1
	P6				1		1	
	P7	1	1	1				
	P8						1	1
	P9	1	1	1		1	1	
	P10	2			1	1		
	P11				1	1		
	P12						1	1
	P13			1		1	1	1
	P14			3				
	P15	1		1	1			1
	P16	2	1	1	1		1	1
	P17	1	1	1	1		1	1
	P18				1	1	1	
mean		1.33	0.72	1.11	0.67	0.28	0.61	0.50

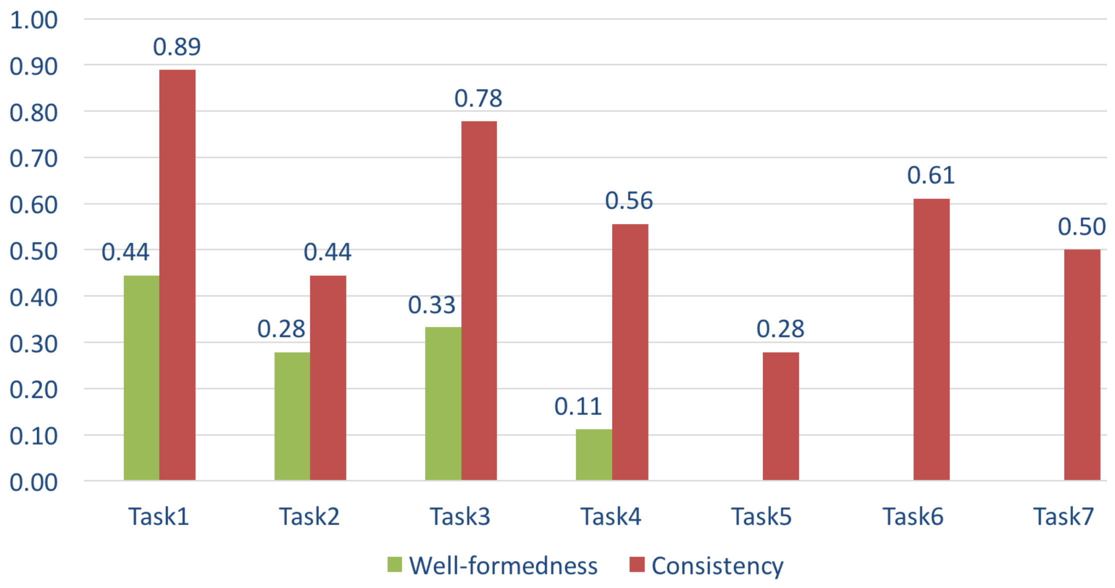


Figure 4.6: Average error rate made by the subjects per task.

Time on Task: Table 4.7 lists the results of the time that each subject took to complete each task. The reason for choosing geometric mean rather than arithmetic mean is the potential skewness that time data may have [7]. As shown in Table 4.7, we can say with 95% confidence that the times on *Task 1* will be between 12.58 and 16.20 minutes.

Fig. 4.7 graphically depicts the average time that subjects took to complete tasks. To ensure that our results are meaningful, the times for 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 to the next task, thus it can dramatically raise the average time on a task.

Tullis and Albert [7] 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. 4.8 shows that the subjects' average times on most tasks were almost twice that of the best achieved times on tasks. Worse, even the most-efficient subject was not as efficient as she could have been as evidenced by the lostness scores, as described below.

Lostness: To estimate the level of lostness, we collected data about how many diagrams or dialogues the subjects visited to perform a particular task. Table 4.8 lists the lostness scores for the subjects on all tasks. It also shows the average lostness score per task. Please note that the average lostness scores in the table refer to data from those subjects who were

Table 4.7: Results of the subjects' time on each task (in minutes).

Participant	Task1	Task2	Task3	Task4	Task5	Task6	Task7
P1	22:10	14:23	08:12	03:49	04:14	02:12	01:34
P2	14:16	07:44	06:17	04:01	04:43	02:10	02:04
P3	07:02	05:58	05:07	06:23	03:58	03:19	06:44
P4	19:43	10:15	10:46	08:03	05:16	02:58	03:03
P5	09:43	05:04	06:24	08:43	05:38	04:08	01:38
P6	16:21	06:40	05:44	08:51	06:04	08:01	07:20
P7	14:39	10:07	05:42	12:36	05:41	06:04	10:30
P8	17:32	08:56	09:57	14:29	06:03	11:48	03:13
P9	14:32	11:28	06:20	04:39	04:25	03:57	04:35
P10	14:18	07:29	09:10	04:10	04:01	04:05	02:29
P11	16:32	13:45	14:40	08:17	04:34	04:21	08:27
P12	14:19	04:43	06:21	08:31	07:30	08:56	05:55
P13	15:36	04:52	09:25	07:11	03:47	03:36	01:52
P14	16:44	06:13	06:02	06:01	06:01	06:13	03:40
P15	19:00	07:10	08:00	04:19	06:25	05:51	02:54
P16	16:46	07:25	04:32	06:36	04:23	04:35	02:23
P17	10:31	04:14	04:49	04:51	03:53	02:46	02:22
P18	12:37	10:28	05:48	06:39	03:31	07:20	01:58
geo-mean	14:39	07:40	07:03	06:37	04:54	04:36	03:23
median	15:08	07:27	06:20	06:37	04:39	04:15	02:59
max	22:10	14:23	14:40	14:29	07:30	11:48	10:30
min	07:02	04:14	04:32	03:49	03:31	02:10	01:34
stddev	03:38	03:02	02:35	02:55	01:07	02:34	02:39
confidence interval (95%)	01:41	01:24	01:12	01:21	00:31	01:11	01:13
upper bound with conf. int.	16:20	09:03	08:14	07:58	05:25	05:47	04:36
lower bound with conf. int.	12:58	06:16	05:51	05:17	04:23	03:25	02:10

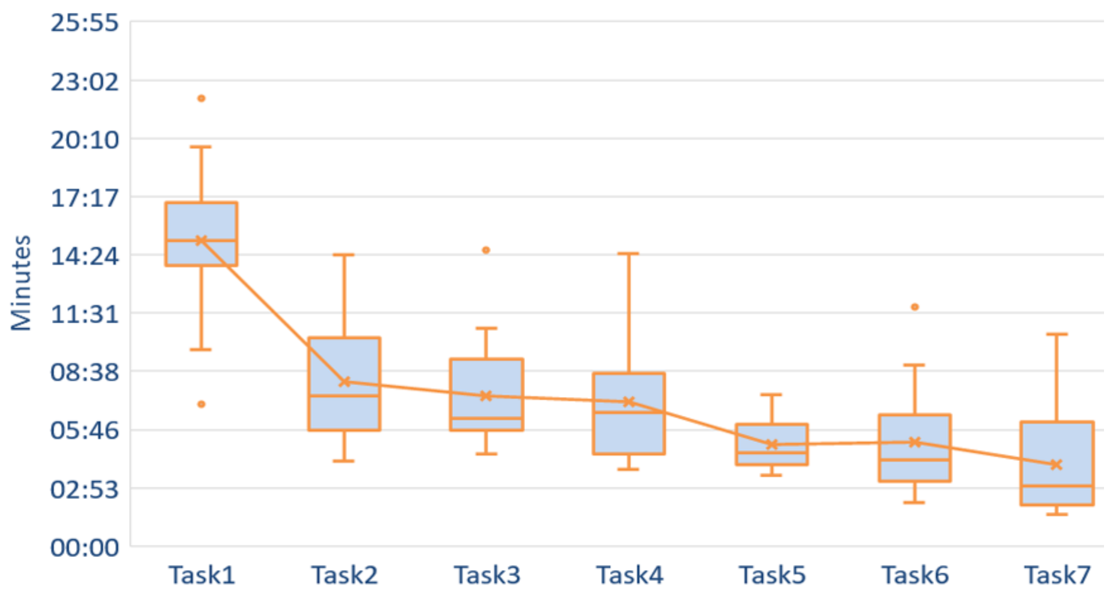


Figure 4.7: The average time to perform each task.

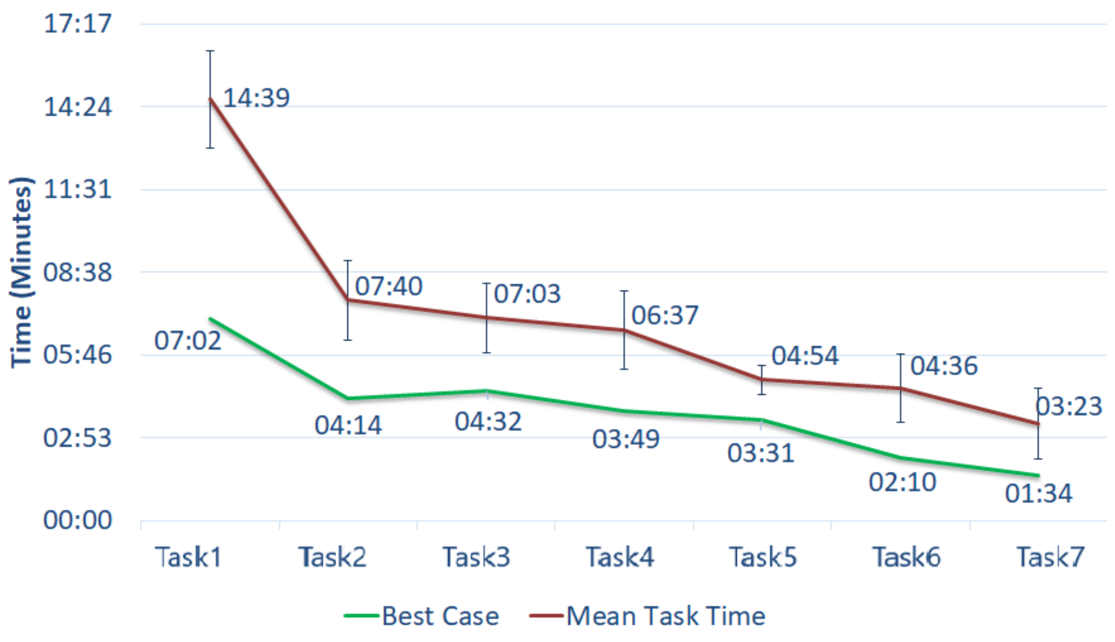


Figure 4.8: Mean time per task vs. best achieved time per task.

completely or partially successful only as it is a more meaningful score than the average of all the scores including unsuccessful tasks. Also, the R value (which is the minimum number of diagrams that should be viewed to complete a task) differ among the subjects. The reason is that, we set the R values based on the tool that was used by the subjects. For example, in some tools the user should open a dialogue box by which they can edit the transition, whereas in some tools the user can edit the transition's label only by clicking on the transition edge.

Table 4.8: Lostness results for the tasks for 18 subjects.

Participant	Task1		Task2		Task3		Task4		Task5		Task6		Task7	
	R,N,S	Lostness	R,N,S	Lostness	R,N,S	Lostness	R,N,S	Lostness	R,N,S	Lostness	R,N,S	Lostness	R,N,S	Lostness
P1	2,3,6	0.60	2,3,3	0.33	2,3,5	0.52	1,2,3	0.60	3,6,6	0.50	1,1,2	0.50	2,1,1	1.00
P2	2,3,4	0.42	2,3,5	0.52	2,4,5	0.54	1,2,4	0.71	3,4,4	0.25	1,2,2	0.50	2,1,1	1.00
P3	1,2,2	0.50	1,2,3	0.60	1,2,2	0.50	1,2,9	0.92	3,3,5	0.40	1,1,1	0.00	2,3,6	0.60
P4	2,3,5	0.52	2,3,5	0.52	2,3,8	0.71	1,2,3	0.60	3,4,5	0.32	1,2,3	0.60	2,2,2	0.00
P5	1,3,4	0.71	1,2,3	0.60	2,2,2	0.00	1,2,7	0.87	3,6,6	0.50	1,1,1	0.00	2,1,1	1.00
P6	1,3,7	0.88	1,2,2	0.50	1,2,4	0.71	1,2,5	0.78	3,5,8	0.55	1,3,6	0.83	1,3,6	0.83
P7	1,2,2	0.50	1,3,5	0.78	1,2,4	0.71	1,3,11	0.99	3,6,7	0.52	1,2,3	0.60	1,2,8	0.90
P8	1,2,4	0.71	1,2,3	0.60	1,2,6	0.83	1,2,6	0.83	3,5,5	0.40	1,2,2	0.50	1,2,4	0.71
P9	1,2,4	0.71	1,2,4	0.71	1,2,3	0.60	1,2,5	0.78	3,5,5	0.40	1,1,1	0.00	1,2,2	0.50
P10	1,2,6	0.83	1,2,5	0.78	1,2,5	0.78	1,2,3	0.60	3,4,4	0.25	1,2,2	0.50	1,2,2	0.50
P11	1,2,4	0.71	1,2,2	0.50	1,2,6	0.83	1,2,6	0.83	3,5,5	0.40	1,2,6	0.83	1,2,10	0.94
P12	1,2,2	0.50	1,3,3	0.67	1,2,2	0.50	1,2,5	0.78	3,6,12	0.71	1,2,11	0.96	1,2,8	0.90
P13	1,3,8	0.91	1,2,2	0.50	1,2,6	0.83	1,2,7	0.87	3,3,3	0.00	1,1,1	0.00	1,1,1	0.00
P14	1,3,12	1.00	1,3,6	0.83	1,3,7	0.88	1,3,5	0.78	3,6,6	0.50	1,3,4	0.71	1,2,3	0.60
P15	1,3,4	0.71	1,3,4	0.71	1,3,7	0.88	1,3,4	0.71	3,6,8	0.56	1,3,4	0.71	1,1,1	0.00
P16	1,3,10	0.97	1,3,6	0.83	1,2,4	0.71	1,2,5	0.78	3,5,6	0.43	1,2,3	0.60	1,1,1	0.00
P17	1,3,7	0.71	1,2,4	0.71	1,2,5	0.88	1,2,5	0.71	3,5,5	0.56	1,1,1	0.71	1,2,3	0.00
P18	1,3,6	0.83	1,3,10	0.97	1,3,7	0.88	1,3,6	0.83	3,1,1	2.00	1,2,5	0.78	1,2,2	0.50
mean (successful subj.)		0.68		0.62		0.65		0.78		0.42		0.57		0.74
efficient subj. (<0.5)		6.67%		7.14%		6.67%		6.67%		40.00%		16.67%		0.00%
inefficient subj. (>= 0.5)		93.33%		92.86%		93.33%		93.33%		60.00%		83.33%		100.00%

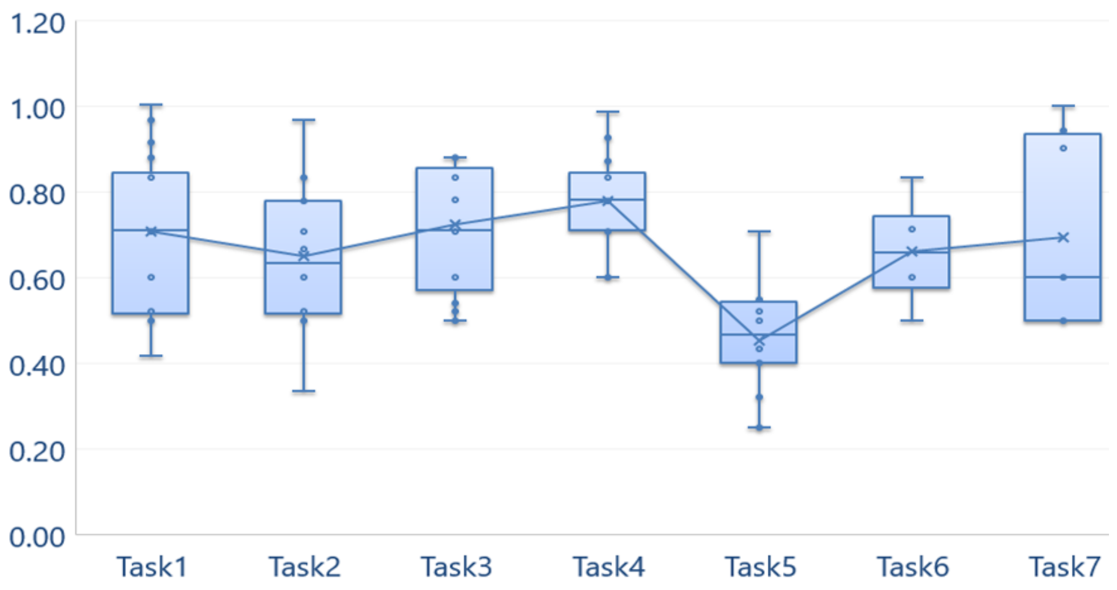


Figure 4.9: The average lostness score for each task.

Fig. 4.9 illustrates the average lostness scores for the subjects. The incomplete tasks are not included in the lostness scores because subjects were not "lost" when they left tasks incomplete. Using Smith's [101] threshold for lostness scores of 0.4, we see that, on average, the only task that the subjects on average performed with a fairly acceptable lostness score was *Task 5*, which was related renaming an element in the Class diagram and propagate the name change to the diagrams that are referring the renamed element, if any. In all other tasks, the subjects on average exceeded the acceptable lostness score which suggests the tools' interfaces are inefficient.

4.4.3 Satisfiability of Users' Expectations (RQ.3)

RQ.3: How well do modelling tools meet users' expectations?

To answer RQ.3, we compared pre-task expectations (see Table 4.9) against post-task experiences (see Table 4.10). The results indicate 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. 4.10 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

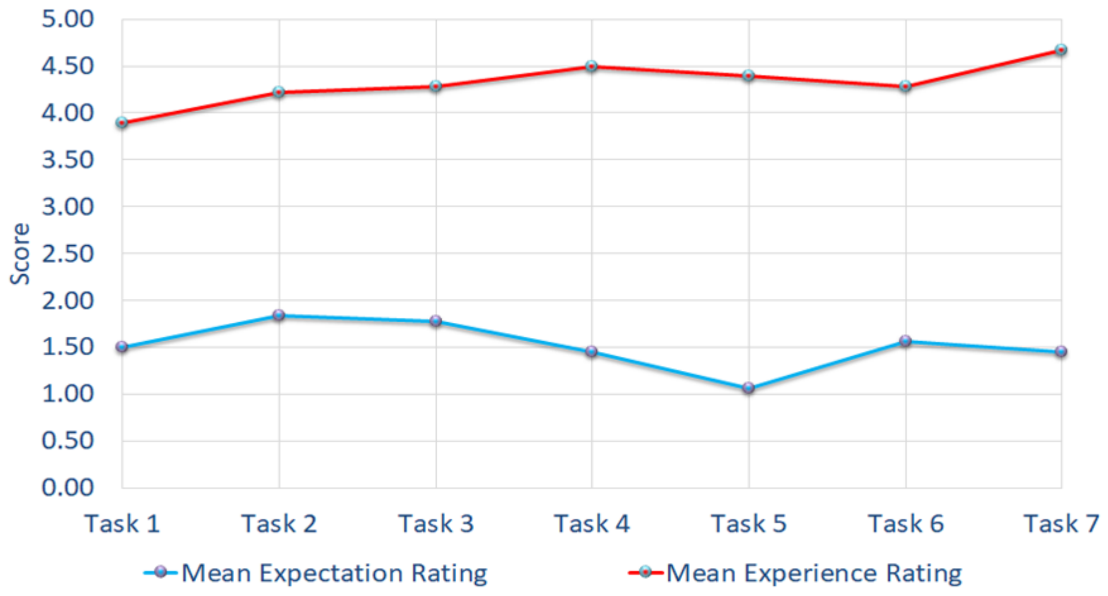


Figure 4.10: Pre- and post-task ratings for each pre-study challenge.

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.

4.4.4 Users' Satisfaction (RQ.4)

RQ.4: Overall, how satisfied are users with modelling tools?

We used CSUQ to measure the users' satisfaction with respect to the 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 Agree) with the neutral value of 4 (see Table 4.11). Fig. 4.11 shows that subjects were slightly dissatisfied with the tools' usability (SYSUSE) and the interface quality (INTERQUAL) (approximately, a mean value of 3), and were more strongly dissatisfied with the tools' ability to provide

Table 4.9: Pre-task expectation ratings.

Participant	Task1	Task2	Task3	Task4	Task5	Task6	Task7
P1	2	3	2	3	1	1	1
P2	2	2	2	1	1	2	1
P3	1	2	1	1	1	1	2
P7	1	1	1	1	1	1	2
P5	1	2	3	2	2	3	2
P4	2	2	2	2	1	2	2
P6	2	1	1	1	1	2	1
P8	1	1	1	1	1	2	2
P9	2	2	1	1	1	1	1
P10	1	2	2	1	1	1	1
P11	1	1	1	1	1	1	1
P12	1	2	2	2	1	2	1
P13	1	2	3	1	1	1	1
P14	2	2	2	1	1	1	1
P15	2	2	2	2	1	2	2
P16	2	2	2	2	1	2	2
P17	2	3	2	2	1	2	2
P18	1	1	2	1	1	1	1
mean	1.50	1.83	1.78	1.44	1.06	1.56	1.44
median	1.50	2.00	2.00	1.00	1.00	1.50	1.00
min	1.00	1.00	1.00	1.00	1.00	1.00	1.00
max	2.00	3.00	3.00	3.00	2.00	3.00	2.00
stdev	0.51	0.62	0.65	0.62	0.24	0.62	0.51
confidence interval (95%)	0.24	0.29	0.30	0.28	0.11	0.28	0.24
upper bound with conf. int.	1.74	2.12	2.08	1.73	1.16	1.84	1.68
lower bound with conf. int.	1.26	1.55	1.48	1.16	0.95	1.27	1.21

Table 4.10: Post-task experience ratings.

Participant	Task1	Task2	Task3	Task4	Task5	Task6	Task7
P1	6	7	5	2	6	3	6
P2	4	3	4	5	4	3	3
P3	2	2	2	2	5	3	7
P7	5	5	3	3	1	4	3
P5	2	5	5	5	5	5	4
P4	3	3	2	3	2	6	6
P6	1	2	3	4	3	2	3
P8	4	4	4	5	7	7	7
P9	5	6	6	5	5	5	5
P10	6	6	7	6	7	7	7
P11	1	2	3	4	2	4	4
P12	1	2	3	4	2	3	3
P13	6	4	5	6	2	1	1
P14	5	4	5	6	5	4	5
P15	6	6	6	6	5	6	5
P16	6	6	5	6	6	5	6
P17	5	5	6	6	7	6	6
P18	2	4	3	3	5	3	3
mean	3.89	4.22	4.28	4.50	4.39	4.28	4.67
median	4.50	4.00	4.50	5.00	5.00	4.00	5.00
min	1.00	2.00	2.00	2.00	1.00	1.00	1.00
max	6.00	7.00	7.00	6.00	7.00	7.00	7.00
stdev	1.94	1.63	1.49	1.42	1.94	1.71	1.75
confidence interval (95%)	0.89	0.75	0.69	0.66	0.90	0.79	0.81
upper bound with conf. int.	4.78	4.97	4.96	5.16	5.29	5.07	5.47
lower bound with conf. int.	2.99	3.47	3.59	3.84	3.49	3.49	3.86

the relevant or the contextual information during the tasks (mean value of around 2.5 for the information quality (INFOQUAL)). The box-plot shown in Fig. 4.11 also shows the median of subjects' ratings. For example, the median of the ratings for the OVERALL category is equal to 2, which denotes that half of the subjects rated the overall quality of the tools greater than or equal to 2 and half of the subjects rated less than 2. Moreover, the short box-plots for SYSUSE, INFOQUAL, and OVERALL suggest that there was a relatively high level of agreement among the subjects with respect to tools' usefulness, information quality, and overall Satisfiability, whereas the tall box-plot for INTERQUAL indicates that the subjects had different opinions with respect to the interface quality of the tools. The long upper whiskers for the box-plots indicate that the ratings were varied amongst the most-satisfied subjects, whereas the short lower whiskers indicate more similar opinions for the least-satisfied subjects.

Table 4.11: Results of the CSUQ questionnaire.

Participant	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19
P1	1	2	1	1	1	2	4	4	1	1	1	1	1	1	3	1	1	1	1
P3	6	7	4	4	4	5	7	6	1	1	5	6	6	5	6	6	6	6	6
P2	2	2	3	1	2	5	5	3	1	1	1	1	2	2	5	5	4	2	2
P7	2	2	2	1	2	1	1	1	3	1	4	3	4	4	2	3	4	4	2
P5	2	4	3	6	3	5	7	5	1	2	2	3	5	4	7	3	3	2	4
P4	4	5	6	4	5	4	7	7	3	3	2	5	N/A	3	5	6	7	5	6
P6	5	3	3	4	3	5	2	3	6	3	6	6	6	5	6	6	6	6	6
P8	2	2	2	3	2	4	7	4	1	N/A	N/A	N/A	N/A	N/A	N/A	4	4	2	3
P9	2	2	2	1	1	1	1	2	1	1	1	2	2	1	2	1	1	N/A	2
P10	1	3	1	1	1	2	5	1	1	1	2	2	N/A	1	1	3	3	1	1
P11	4	6	7	6	4	4	N/A	N/A	5	7	N/A	6	N/A	N/A	7	6	6	1	6
P12	3	4	4	3	3	4	3	4	1	2	4	3	5	3	4	5	5	1	4
P13	1	2	3	1	1	1	2	1	1	1	1	1	1	1	1	4	1	1	1
P14	2	4	5	1	1	2	3	4	1	1	1	3	2	1	5	3	2	1	3
P15	2	3	1	3	2	2	5	2	1	1	2	2	N/A	N/A	N/A	3	2	2	2
P16	1	1	2	4	4	2	2	3	1	1	1	5	N/A	N/A	5	4	4	3	2
P17	3	3	4	4	3	2	1	3	1	2	4	5	5	3	5	4	4	1	2
P18	1	1	5	1	1	1	6	2	1	N/A	N/A	1	N/A	6	3	1	1	7	1
mean	2.44	3.11	3.22	2.72	2.39	2.89	4.00	3.24	1.72	1.81	2.47	3.24	3.55	2.86	4.19	3.78	3.56	2.71	3.00
median	2.00	3.00	3.00	3.00	2.00	2.00	3.50	3.00	1.00	1.00	1.50	3.00	1.50	1.50	4.50	4.00	4.00	2.00	2.00
min	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	1.00
max	6.00	7.00	7.00	6.00	5.00	5.00	7.00	7.00	6.00	7.00	6.00	6.00	6.00	6.00	7.00	6.00	7.00	7.00	6.00
OVERALL										2.98									
SYSUSE										2.99									
INFOQUAL										2.79									
INTERQUAL										3.36									

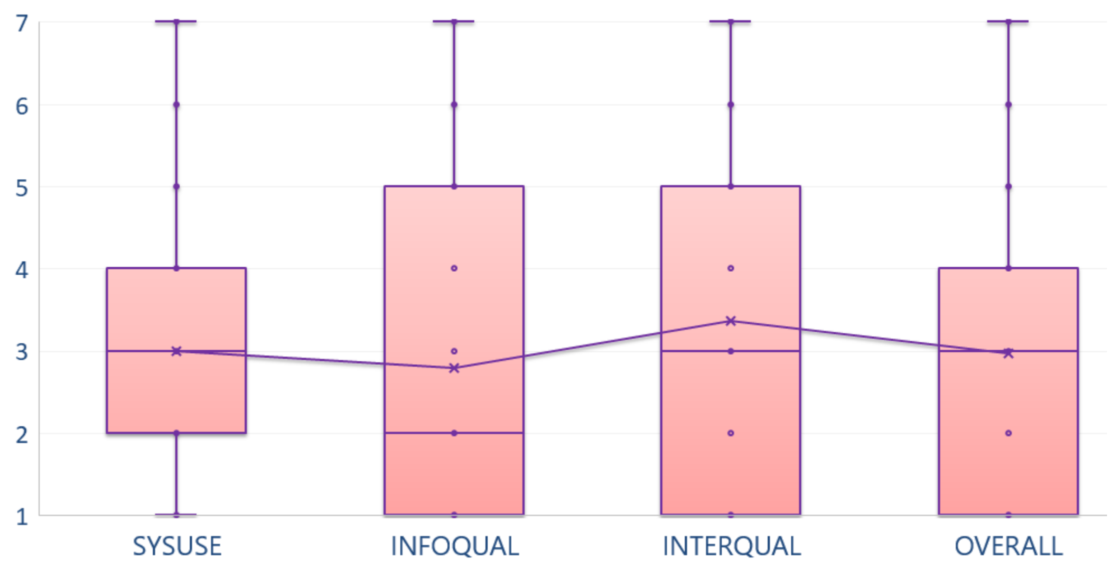


Figure 4.11: Box-plot based on the result from CSUQ.

4.4.5 The Most-Severe Challenges (RQ.5)

RQ.5: Which challenges are the most severe experienced by modellers employing modelling tools?

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 results from the three analyses support one another.

4.4.5.1 Pre- and Post-Session Ratings

We used pre- and post-session ratings (see Tables 4.12 and 4.13, respectively) to identify pre-study challenges that users expected the tools to alleviate, but that the tools did not. Fig. 4.12 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 would improve their tools the

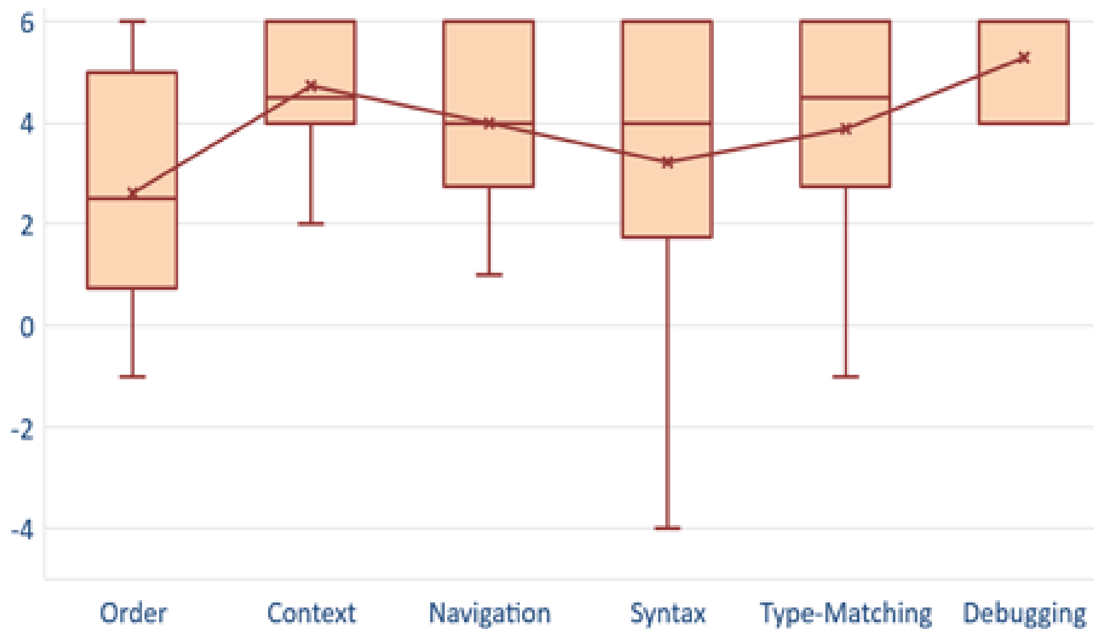


Figure 4.12: Mean discrepancy between the pre- and post-session ratings for each pre-study challenge.

most by proposing tool advances that address these two prominent challenges over other tool advances.

Table 4.12: Pre-session expectation ratings.

Participant	Order	Context	Navigation	Syntax	Type-Matching	Debugging
P1	1	1	2	7	2	1
P2	2	1	3	1	1	1
P3	1	1	1	1	1	1
P7	2	1	1	1	1	2
P5	1	1	3	4	3	1
P4	1	2	1	2	1	1
P6	2	2	2	1	1	1
P8	1	1	1	1	1	1
P9	2	2	1	3	1	1
P10	1	1	1	1	1	1
P11	2	1	1	1	1	1
P12	2	1	2	3	2	1
P13	1	1	1	1	1	1
P14	2	1	2	1	2	2
P15	2	1	1	1	2	2
P16	3	1	1	2	2	1
P17	1	2	2	2	3	1
P18	1	1	3	1	1	2
mean	1.56	1.22	1.61	1.89	1.50	1.22
median	1.50	1.00	1.00	1.00	1.00	1.00
min	1.00	1.00	1.00	1.00	1.00	1.00
max	3.00	2.00	3.00	7.00	3.00	2.00
stdev	0.62	0.43	0.78	1.57	0.71	0.43
confidence interval (95%)	0.28	0.20	0.36	0.72	0.33	0.20
upper bound with conf. int.	2.90	4.92	4.36	3.95	4.22	5.48
lower bound with conf. int.	2.33	4.52	3.64	2.50	3.56	5.08

Table 4.13: Post-session experience ratings.

Participant	Order	Context	Navigation	Syntax	Type-Matching	Debugging
P1	6	7	7	3	7	7
P2	5	7	5	7	7	7
P3	2	6	7	7	7	7
P7	4	3	5	6	6	6
P5	4	7	5	5	2	6
P4	6	6	4	4	6	5
P6	2	6	3	5	4	7
P8	7	7	7	7	7	7
P9	7	6	7	7	7	7
P10	7	7	7	7	7	7
P11	1	4	3	1	2	6
P12	1	5	6	5	4	5
P13	1	5	5	7	6	7
P14	4	7	6	4	4	7
P15	4	4	4	5	5	6
P16	6	7	7	5	6	7
P17	6	6	7	6	6	7
P18	2	7	6	1	4	6
mean	4.17	5.94	5.61	5.11	5.39	6.50
median	4.00	6.00	6.00	5.00	6.00	7.00
min	1.00	3.00	3.00	1.00	2.00	5.00
max	7.00	7.00	7.00	7.00	7.00	7.00
stdev	2.20	1.26	1.42	1.94	1.69	0.71
confidence interval (95%)	1.02	0.58	0.66	0.89	0.78	0.33
upper bound with conf. int.	3.63	5.30	4.66	4.12	4.67	5.60
lower bound with conf. int.	1.59	4.14	3.34	2.33	3.11	4.95

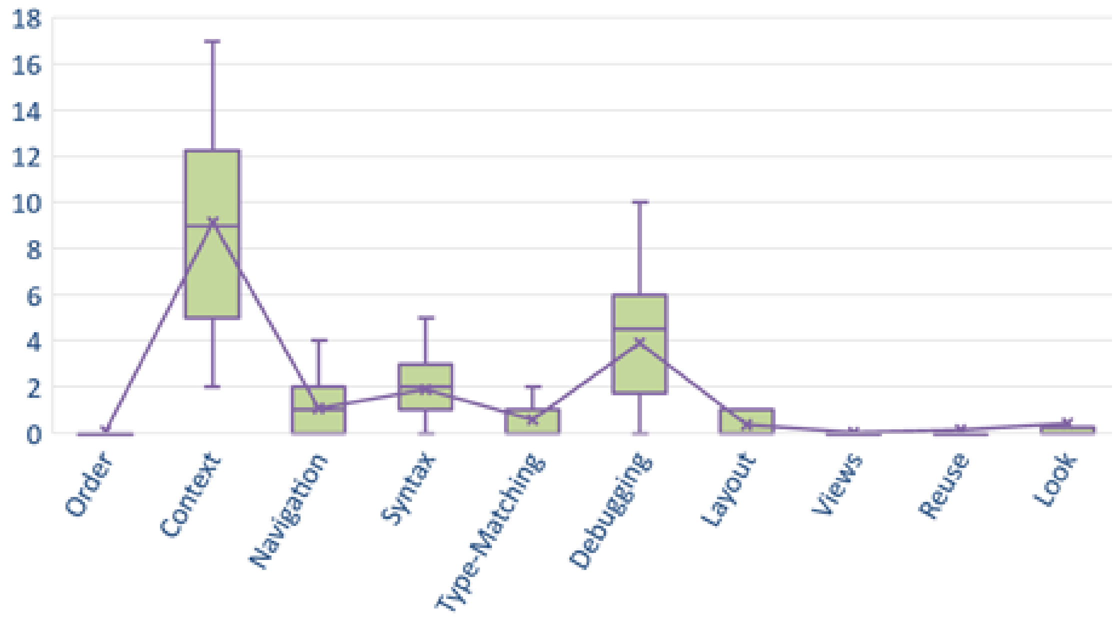


Figure 4.13: Mean frequency of each challenge expressed verbally by each subject.

4.4.5.2 Behavioural Metrics

The think-aloud protocol also helped us to identify 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 results of our verbal analysis is shown in Table 4.14.

As shown in Fig. 4.13, 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…”*. Table 4.15 shows a complete list of statements that we coded for one of the subjects.

Table 4.14: Results of the analysis and coding of the verbal expressions.

Participant	Order	Context	Navigation	Syntax	Type-Matching	Debugging	Layout	Views	Reuse	Look
P1	0	2	0	5	1	5	0	0	0	0
P3	0	5	1	1	0	4	0	0	0	0
P2	0	4	1	1	0	6	0	0	0	0
P7	0	5	0	2	0	5	0	0	0	0
P5	0	4	1	0	2	5	1	0	0	0
P4	0	5	0	2	0	0	0	1	0	0
P6	0	14	2	5	1	8	0	0	1	0
P8	0	11	3	0	5	6	3	0	1	0
P9	1	9	1	3	1	6	0	0	0	1
P10	1	9	0	2	0	2	0	0	0	1
P11	0	17	4	1	0	2	0	0	0	0
P12	0	8	0	3	1	1	1	0	0	0
P13	0	9	1	1	0	2	1	0	0	0
P14	0	12	0	1	0	1	0	0	0	0
P15	0	11	0	0	0	2	0	0	0	0
P16	0	13	2	2	0	10	0	0	0	0
P17	0	17	2	2	0	5	1	0	0	4
P18	0	10	1	3	0	0	0	0	1	2
sum	2	165	19	34	11	70	7	1	3	8
mean	0.11	9.17	1.06	1.89	0.61	3.89	0.39	0.06	0.17	0.44
median	0.00	9.00	1.00	2.00	0.00	4.50	0.00	0.00	0.00	0.00
min	0.00	2.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
max	1.00	17.00	4.00	5.00	5.00	10.00	3.00	1.00	1.00	4.00
stdev	0.32	4.44	1.16	1.49	1.24	2.81	0.78	0.24	0.38	1.04

4.4.5.3 Analysing Errors

The error rates presented in Fig. 4.6 show that almost all of 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. 4.5 show a high degree of failure in the subjects' average performance for these tasks.

Based on the above results, we conclude that the *Context* and *Debugging* challenges are the most-severe challenges. This conclusion is confirmed by other metrics such as the frequency with which subjects switched to other diagrams during tasks, more specifically the Class diagram, as a reference metric for the severity of the *Context* challenge.

4.5 Threats to Validity

In this section, we discuss the most important threats to the validity of the study and provide suggestions for improvements in future studies of this type.

4.5.1 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.

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. 4.8) suggests that the tools do not provide enough useful and usable features for such debugging tasks.

Table 4.15: A list of statements coded for an exemplary subject.

Category	Phrase
Context	It hurts me that the tool does not tell me what is each object. For example, I see g as a variable, but it would be naive if it gives me some message that g is an object of Gate when I hover on it.
Syntax	Also, syntax highlighting would be really nice. It doesn't highlight keywords.
Syntax	I would love if I press control space and it brings me the syntax. This is a wasting time to write after ! I would love to just write a and it filters after for me and show it to me. Some how it reduces my performance.
Syntax	Also, it would be nice if it gives me what's the unit . But I know that it is in milliseconds. It would be nice if I hover on after it tells me that "OK, for after every 1000 is a one second").
Syntax,Debugging	Another problem that I have right now, is that my text editor doesn't show that my syntax is wrong, I need to compile. I don't like this one. I want immediate actions from my IDE.
Context	Right now , Sensor is in the class diagram and it is hard to manage. (referring to textual presentation), because the class diagram is huge.
Context	Another problem is that this tool doesn't allow me to see the graphical representation, so it is gonna be really hard for me.
Reuse	This is really lacking graphical representation
Context	I would love to see the attribute coming up here. I don't see the name of attributes or operations.
Context	I don't remember. I'm gonna go to class diagram , I'm gonna copy this one, the name of attribute and come to this one (diagram) and put it here.
Context	I'm loving this one because I'm just copying and pasting the model
Context	Right now I don't remember what is the attribute
Context,Navigation	Right now, I don't know how to get to the coin slot! Ahhh! Damn it! I wish I had the graphical representation for this one.
Context	It would be really nice that when I click on the name (of the variable) it find me to the class gate and show me the properties of it. It would be really good, I'm missing that one here!
Context	I can't have a blueprint of the system. I don't see any association.
Context	I wish it would bring me the name here !
Debugging	That's an issue , it doesn't do realtime checking of my model.
Context	remembering the contextual information was hard to me
Context	I had issue with switching between classes [to remember sth]
Navigation	(C3) I have some issue with the exitfee, but probably I would have issue if I wanted to design from the beginning.
Syntax	(C4)The tool didn't allow me to have that one in a fast way. E.g., writing half of the keyword and the rest should be written automatically.
Debugging	The tool didn't allow me to find errors in the system. Actually it detected the error in the system, that was accurate, but the hint for the location of the error was not accurate.
Debugging	I know the tool is lacking this one (auto-propagating the change), that's why I'm doing that (finding manually).
Debugging	I would like to see that when I'm changing that one it pops up options for me and ask that "O1 do you wanna apply it to whole model or do it manually?". So it doesn't do it. So I had to do tricks based on my experience to solve the issue.
Debugging	It is critical if the model is big.
Context	let me check the class diagram again.
Debugging	System didn't detect that we don't have this attribute. It didn't detect but I checked visually (by my eyes).
Debugging	I'm assuming that these are correct ! It doesn't show me any errors.
Type-Matching	I know the tool doesn't check the validity of the LHS and RHS, I wanted to makes sure the position is string or not.

4.5.2 Internal Validity

One threat to internal validity was the subjects' familiarity with the UML. We tried to mitigate against this risk by recruiting subjects who could pass 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 their levels of competence in UML modelling.

4.5.3 External Validity

The main threat to external validity is that the subjects were students rather than experienced modellers who work in industry. To mitigate against this threat we kept the scope of the study and the size of the model quite small compared to industrial models of software systems.

The size of the model was not comparable to the large-scale complex systems one would expect to find in industry (e.g., see [56]). 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 subjects would have been more familiar with the model.

4.6 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 a model expresses the same information as the experiment's textual description (i.e., correctness errors are not methodological errors).

- Some tools such as MagicDraw [45] or Visual Paradigm [75] allow users to cross-link operations in the event/effects of a transition expression 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.

In general, these modelling-tool challenges stem primarily from the separation of concerns that are inherent in the distinct views captured in distinct UML diagrams. UML modelling tools offer diagram-specific editors that do little to help users fetch and understand the inter-related information that is expressed separately in other related diagrams [47]. For example, developing a correct expression for a state-transition guard in a State-Machine diagram can be cumbersome for the modeller because they need to refer to precise details about names, types, attributes, parameters, and associations of model elements that are defined in a separate Class diagram.

4.7 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 of 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 of the tools' usability leaned towards dissatisfaction.

In the next chapter we propose enhancements to tools that address these most-critical challenges. For each challenge, we identify relevant human-cognition factors that might effectively reduce the challenge and devise enhancements to the tools that reinforce the identified factors. In Chapter 6, we present the results of empirical user studies that assess the impact of the tool enhancements on modellers' effectiveness.

Chapter 5

UCAnDoModels

"...it is likely that you have purchased a variety of artifacts, such as calendars, calculators, or computers, that were created primarily to support cognitive activities... The vast majority of these cognitive artifacts were designed not by the application of cognitive theory but instead by appeal to folk-psychological intuitions, trial and error, and the forces of the marketplace. Only recently have researchers begun to consider how theories and findings from cognitive science can be systematically applied to the design of the cognitive environment."

— – Alex Kirlik, *"Everyday Life Environments"[50]*

We propose **U**ser-**C**entric and **A**rtefact-Ce**n**tric **D**evelopment **o**f **Models** (*UCAnDoModels*), which employs both user-centric and artefact-centric strategies to overcome users' foremost challenges with UML model editors and we built a model editor that incorporates the UCAnDoModels approach. The main philosophy behind our model editor is to provide the modeller with sufficient contextual information that is relevant to performing a task, by offering contextual interfaces, semi-automated and on-the-fly repairs, and auto-completions. This chapter[6] presents the core components of our editor and their underlying capabilities.

The rest of this chapter is organized as follows. At first, we present the architecture of our tool, its components and their interactions. Afterwards, we discuss each component

[6]This chapter is reprinted in adopted form from [81] and [82]

starting with some low-level features that are used in multiple places, namely Distance-Oriented Object Indexer, Type-Based and Distance-Oriented Content-Assist, Distance-Based Model Element Finder and Model Observer. Then, we present our auto-completion technique followed by the interfaces that we proposed to reduce the Context and Debugging challenges of modellers. We start the description of the interfaces by discussing the Focus+Context Transition Editor and the Interactive Consistency Management Interface, which are the most-important interfaces we proposed to reduce the efforts of editing transitions and maintaining consistencies in a model. Then, we explain the rest of the interfaces.

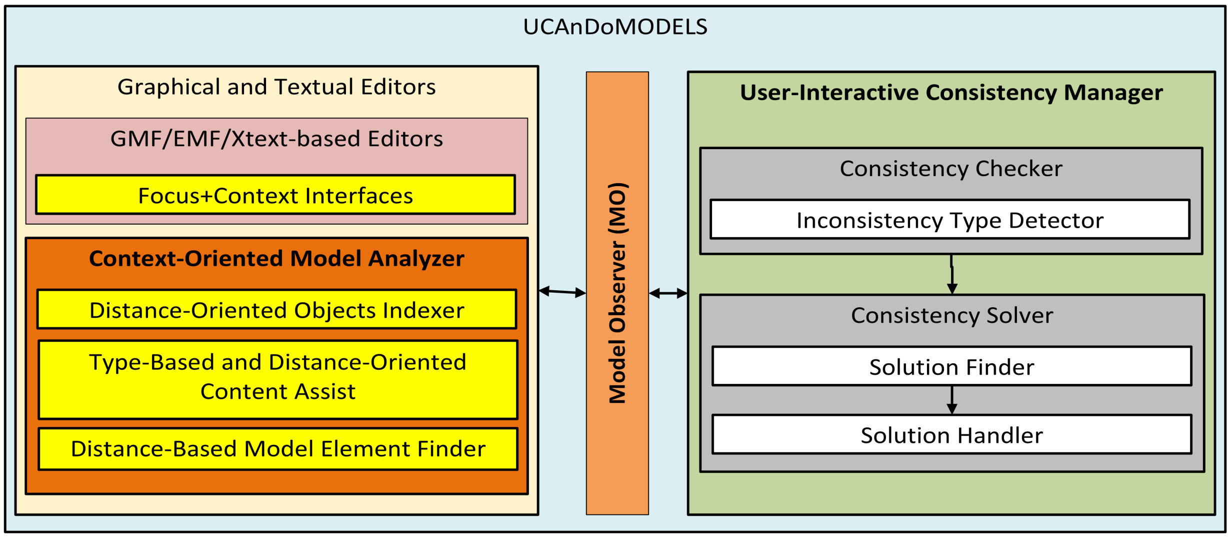

UCAnDoMODELS
Graphical and Textual Editors
GMF/EMF/Xtext-based Editors
Focus+Context Interfaces
Context-Oriented Model Analyzer
Distance-Oriented Objects Indexer
Type-Based and Distance-Oriented Content Assist
Distance-Based Model Element Finder
Model Observer (MO)
User-Interactive Consistency Manager
Consistency Checker
Inconsistency Type Detector
Consistency Solver
Solution Finder
Solution Handler

Figure 5.1: The architecture of our UCAnDoModels editor and its components.

5.1 Architecture

The architecture of our tool is shown in Fig. 5.1. The **Graphical and/or Textual Editors** embody the primary components that allow users to edit a model. It consists of default diagramming features as well as our proposed Focus+Context Interfaces. To achieve their goals, the Focus+Context Interfaces also communicate to the Context-Oriented Model Analyzer component, which offers low-level backend-related features to filter, classify, and rank model elements based on their relevance to the focus task. In fact, the Focus+Context Interfaces execute these features to extract valid and relevant contextual information that is relevant to a focus task.

The Graphical and Textual Editors share the state of the model with the **Model Observer (MO)** component whose main task is to listen to the model edits from the Graphical and/or Textual Editors and react accordingly. That is, the MO may perform follow-up actions to a model edit or notify the **User-Interactive Consistency Manager** about inconsistencies introduced to the model as the result of an edit. The User-Interactive Consistency Manager is responsible for maintaining the model's consistency by first detecting the type of the inconsistency based on a pre-defined set of consistency rules (embedded in the Inconsistency Type Detector module), and then offering proper solution(s) to resolve the inconsistency by executing the Solution Finder module. Moreover, if resolving the inconsistency requires the user to intervene, then the Solution Handler passes the inconsistency to the Graphical and Textual Editors, which pops up the corresponding Focus+Context Interface that asks the user to fix the inconsistent element(s).

Our primary contributions are the Focus+Context Interfaces and the Context-Oriented Model Analyzer which are highlighted in yellow in the figure. We implemented our tool from scratch (i.e., not extending an existing open-source model-editor). The essential diagramming features were built using Eclipse's Ecore Modelling Framework (EMF) [11], the Graphical Modeling Framework (GMF) [41], and the Xtext Textual Editor [26].

5.2 Distance-Oriented Objects Indexer

The goal of the Distance-Oriented Object Indexer is to rank the information such that the most-relevant information is presented first to the user, which helps in decreasing the modeller's memory workload. This feature is adapted (with modifications) from existing coding environments.

The Distance-Oriented Object Indexer is responsible for collecting information about the relationships among the model elements declared in the Class diagram. Such information is used in ranking the information (e.g., model elements) presented to the user such that the information deemed most relevant is ranked first. Specifically, whenever the editor's focus changes to a different element, the indexer computes the *Distance* value between each model element and the specific element that is the current focus of an editing task. For example, if we are writing a transition expression in a State-Machine diagram for class C1, the Distance-Oriented Objects Indexer categorizes the model's elements into five categories as follows:
Category A: comprises the values or literals of the type of the model elements under focus (i.e., basic types such as Boolean, Integer and user-defined types such as enumerations). For instance, if the model element of current focus is of type Boolean, then values *True*

and *False* are included in this category. Basic-type values or literals have a *Distance* of 0.
Category B: comprises the object members (e.g., operations or attributes) that are defined in class C1. The distance for an element in Category B is one ($Distance = 1$).
Category C: comprises the model elements that are defined in the ancestor classes of C1. The elements in this category have a *Distance* of 2.
Category D: comprises the model elements that are defined in the classes with which C1 has a direct relationship (association, composition or aggregation). The *Distance* for a class related by composition or aggregation is equal to 3 and the *Distance* for a class related by an association is 4. Thus, compositions have priority over associations.
Category E: comprises the model elements that are defined within the classes that are indirectly related to C1. The *Distance* for an element in this category depends on the number of links (edges) between the element and class C1; that is $Distance = 4 + (NumberOfLinks - 1)$. Thus, the further the element, the less related it is.

The Indexer then ranks elements/relationships based on their respective *Distance* values and alphabetically sorts the elements/relationships that have the same *Distance*, which reduces the user-tool interaction time (i.e., U).

5.3 Type-Based and Distance-Oriented Content-Assist

The Type-Based and Distance-Oriented Content-Assist aims at proposing the user a list of model elements (or language keywords) that are syntactically and semantically valid. The Content-Assist feature decreases the modeller's short-term and long-term memory workload and decreases number of errors due to lapses or knowledge-based mistakes. The *Type-Based* portion of the feature is adopted from existing tools (e.g., Eclipse), whereas the *Distance-Oriented* portion extends the existing Content-Assists. The following explains our Type-Based and Distance-Oriented Content-Assist.

One of the challenges of designing a Content-Assist is proposing relevant and valid options to the user [77]. The existing editors retrieve the list of available options from the meta-model and its constraints. They essentially populate the Content-Assist list based on the language syntax rather than analysing the model elements and filtering out the ones that are not semantically sound. In contrast, our editor provides **Type-Based and Distance-Oriented** Content-Assist to propose values or model elements that are most likely to be the intended next token or clause in the expression currently under edit, thereby reducing the effort of editing the expression. Our Content-Assist uses the

Distance-Oriented Objects Indexer component described above and embeds a *Type-Based Classifier* that performs lightweight type checking to ensure that variable assignments and conditional expressions (e.g., guard conditions) are semantically sound by type. For instance, if an operand in a condition is an attribute of type T1, then the other operand should be limited to attributes or operations with the return type of T1 or sub-type of T1. Moreover, if T1 is from the type of Boolean, then the other operand itself can be a conditional expression which can be evaluated to *True* or *False*. After filtering out the elements that are not sound-by-type, the Distance-Oriented Objects Indexer ranks and orders the elements. Please note that, this process of filtering out the elements is performed based on the current expression that the Content-Assist is being populated. For example, if there are nested expressions, we first identify the expression (also referred to as *scope*) that the Content-Assist is being viewed for, and filter our only those elements that violate the type soundness of the expression under edit (i.e., scope).

With our Content-Assist, the user will be given only options that are syntactically correct and semantically sound. Such assistance is expected to improve model correctness and reduce modelling efforts (by reducing user's think time Z and user-tool interaction time U), because the modeller is offered a refined list of valid options rather than an exhaustive list of all syntactically-related model elements, which can be lengthy and difficult to explore.

5.4 Distance-Based Model Element Finder

The Distance-Based Model Element Finder is a new feature that is proposed by us to ease a modeller's task of finding model elements when editing and debugging models. By using the Distance-Based Model Element Finder the modeller does not require to remember the exact name of the intended model element, which indeed reduces modeller's short-term and long-term memory workload.

This module allows users to search for an intended model element among all the existing model elements. It first attempts to find the elements whose names exactly match what the user typed. However, it is not always the case that the user remembers the exact name of an element; a user usually remembers a part of a name or tries to "guess" a similar name. Therefore, if no element with the exact name is found, this module uses the *Levenshtein* edit distance [59] algorithm, an approximate string-matching algorithm, to look for elements whose names are similar to what the user typed and list them in increasing order of the computed edit distance. If multiple elements with the same edit distance are found, the module uses the Distance-Oriented Object Indexer to rank and order them before displaying them to the user. The search is online, which means that the user can see the results as they type. That is, as soon as the user starts typing, the tool looks for the elements that match the user's typed text (i.e., sub-string), and displays the results as a list of elements. The results list also shows the element's abstract type (e.g., attribute, operation, class). For example, if there is a *gateCounter* attribute in the *Gate* class in the Class diagram, and the users types *"count"* in the search box, the results list will include the respective item displayed as *"Gate.gateCounter :: Attribute"*.

The Distance-Based Model Element Finder can reduce the modeller's think time (i.e., Z) as the modeller does not have to remember the exact name of an element, and can instead find an intended model element by only remembering a part of the element's name, leading to a reduced memory workload.

5.5 Model Observer

The **Model Observer (MO)** is a feature that is adapted from existing tools that continuously listens to model edits and reacts with consistency fixes or warnings, or with auto-completions. In fact, together with the User-Interactive Consistency Manager component, their main goals are to facilitate the tasks of maintaining model's consistency and offering auto-completions when editing and debugging models, which decrease execution, knowledge-based and planning-related failures, and reduce short-term and long-term memory workload. Our MO is explained as follows.

5.5.1 Maintaining Consistency

Correct-by-Construction approaches are known to suffer from being intrusive and counterproductive [27][39], which adversely affects users' performance and satisfaction. The philosophy behind our User-Interactive Consistency Management is to provide easier error-resolution techniques for the modelling tools that adhere to Correct-by-Construction. We hypothesize that a Correct-by-Construction approach would perform better if the tool eased the debugging and error-correcting task by employing more interactive techniques such as informative and timely error notifications, automated fixes (where possible), and Focus+Context interfaces for non-automated fixes (see Section 5.6.2).

For consistency management, we use the MO to observe model edits and check that each edit satisfies the editor's consistency rules. The rules that our model editor enforces (shown in Table 5.1) are adapted from *Well-formedness*, *Consistency*, and *Completeness* rules proposed by Lange et al. [55]:

- *Well-formedness*: Rules in this category ensure the syntactic correctness of the model with respect to the language's meta-model. Rules in this category must be satisfied in order to avoid creating diagrams that contain syntactic defects.
- *Consistency*: The rules in this category are used to check for contradictory information among the diagrams or within a single diagram. If rules in this category are not satisfied, the model is semantically defective. For instance, if a transition's event refers to an operation that is not defined in the Class diagram, then this usage of operation would be flagged as an inconsistency.
- *Completeness*: The completeness rules check whether a model element is defined but not used, or if a model element should exist to make the model more meaningful, but it does not exist. If the rules in this category are not satisfied, the model is not defective, but can be improved.

In Table 5.1, the WF1, WF3, WF4, WF5, WF6, WF7, and WF8 Well-formedness rules should not be seen as Completeness violations, as these Well-formedness rules are discouraged in the language's specification, while the Completeness rules are mostly related to providing more complete information to create a more meaningful model.

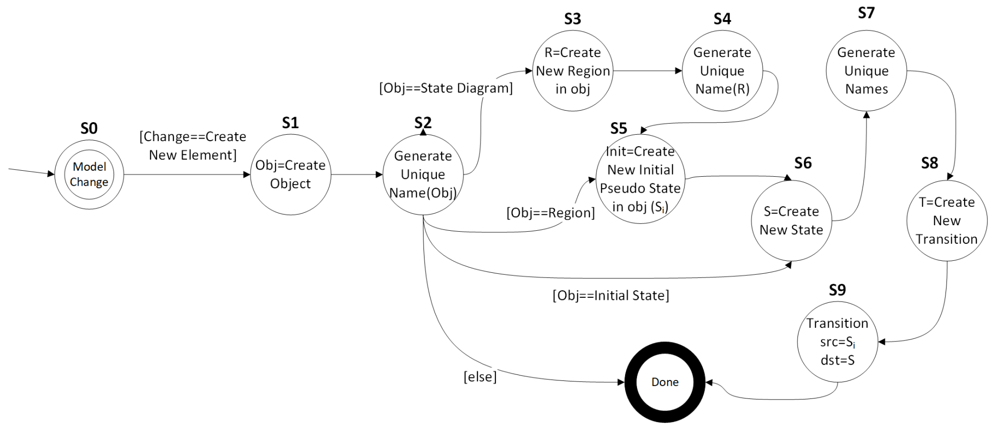
If any of the Well-formedness, Consistency, or Completeness rules are violated as a result of a model edit, the MO notifies the User-Interactive Consistency Manager module, described below.

The User-Interactive Consistency Manager component consists of two main modules: a Consistency Checker and a Consistency Solver. The Consistency Checker detects the *type* of inconsistency based on Lange's definition [55] (note that only the *existence* of an inconsistency is detected by MO and not the *type* of the inconsistency) and finds a proper solution using the Solution Finder and Solution Handler modules (embodied in the Consistency Solver), respectively. A solution can be Auto-Fix, Quick-Fix, or Interactive-Fix. Auto-Fix applies if the Solution Finder suggests that there is only one possible valid fix for an inconsistency (e.g., renaming a model element should rename all its uses in the model); the Solution Handler asks the user to confirm the fix before automatically fixing the model. Quick-Fix applies if the Solution Finder identifies a finite number of valid fixes for an inconsistency. The Solution Handler then proposes the list of possible fixes and allows the user to choose one from the list. For example, if there is an initial pseudo state

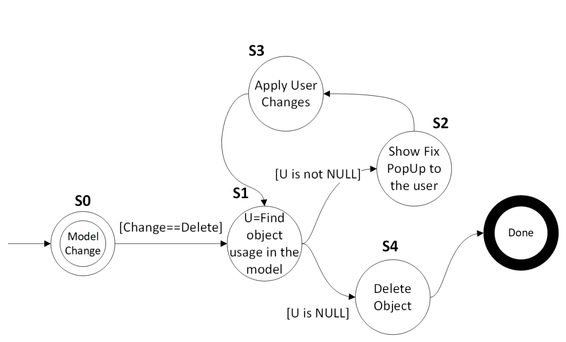
Table 5.1: List of consistency rules.

ID	Type	Description
WF1	Well-formedness	Elements without name
WF2	Well-formedness	Duplicate element names
WF3	Well-formedness	Classes without at least one operation or attribute
WF4	Well-formedness	Transitions without triggering event or guards
WF5	Well-formedness	States without transitions connected to them
WF6	Well-formedness	Initial pseudo states that do not refer to a state
WF7	Well-formedness	Regions without an initial state
WF8	Well-formedness	State diagrams with no regions
CON1	Consistency	Diagrams refer to undefined classes
CON2	Consistency	Triggering events refer to undefined attributes and messages
CON3	Consistency	Transition guards refer to undefined attributes and messages
CON4	Consistency	Transition actions refer to undefined attributes and messages
CON5	Consistency	A message that is used in the model has a wrong returning type (type mismatch)
CON6	Consistency	Mismatch between the nodes in the Feature Model and the feature classes in the Class diagram
CON7	Consistency	An AND/OR/XOR node in a Feature Model that has only one child node (maybe because the other sibling nodes were deleted or moved)
COM1	Completeness	Class defined in class diagram is not used in the model
COM2	Completeness	Operation defined in class diagram is not used in the model
COM3	Completeness	Attribute defined in class diagram is not used in the model
COM4	Completeness	Operation's attributes defined in class diagram are not used in the model
COM5	Completeness	An AND, OR, and XOR node in the Feature Model that has no more than one child node

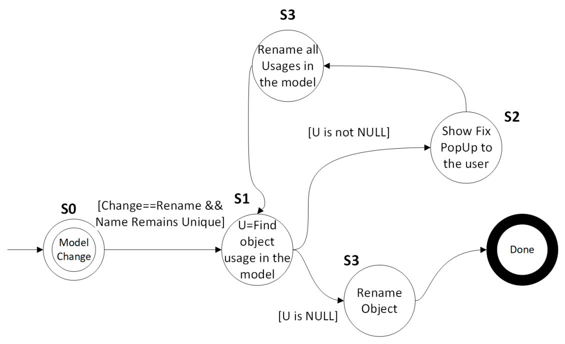
in a region that is not connected to any other state, the Solution Handler provides the user with a list of possible existing states to which the initial pseudo state can be connected. If the Solution Finder decides that neither Auto-Fix nor Quick-Fix techniques are applicable, it displays our Interactive-Fix dialogue and asks the user to intervene to resolve the issue before proceeding with subsequent model edits (described in 5.6.2). Moreover, since we are interested in Correct-by-Construction, the tool does not proceed with the model edits unless the user resolves all the inconsistencies; that is, the user is required to fix the inconsistencies before proceeding with subsequent model edits.



(a) Creating model elements.



(b) Deleting model elements.


S3
Rename all Usages in the model
S2
Show Fix PopUp to the user
[U is not NULL]
S0
Model Change
[Change==Rename && Name Remains Unique]
S1
U=Find object usage in the model
S3
Rename Object
[U is NULL]
Done

(c) Renaming model elements.

Figure 5.2: MO behaviour for different types of actions.

Table 5.2: List of Head and Tail actions.

Head Action	**Tail Action(s)**
Create a new model element	Generate/Set a unique name for it
Rename a model element	Ensure the name remains unique and propagate the new naming to all uses of the element
Create new State-Machine diagram sd_1	Create new region r_1 in sd_1
Create new Region r_1	Create new initial pseudo state in r_1
Create new initial pseudo state s_i in r_1	Create new state s_1 in r_1 and create a transition from s_i to s_1
Create state s_2 inside basic state s_1 that has no inner region	Create a region r_1 in s_1, create an initial pseudo state s_i, move s_2 to r_1, and create a transition from s_i to s_2
Delete a class in Class diagram	Delete the corresponding State-Machine diagram (needs user's confirmation)
Delete a feature class in Class diagram	Delete corresponding State-Machine diagram (needs user's confirmation) and delete corresponding feature node from Feature Model and bring the deleted feature's children to the deleted feature's parent node
Delete a model element in Class diagram that is used in a State-Machine diagram	Fix the inconsistency before proceeding
Use an undefined operation or attribute in a State-Machine diagram	Define the missing operation or attribute in the Class diagram
Move a Class attribute or element to another Class	Update all navigation paths that include the moved element (where possible)
Create a feature node in the Feature Model	Create the corresponding feature class in the Class diagram
Delete a feature node in the Feature Model	Delete the corresponding feature class from the Class diagram and its corresponding State-Machine diagram (asks user's confirmation if the feature is used in other parts of the model), assign (move) its children feature nodes to its parent feature node
Create a feature class in the Class diagram	Create a feature node in the Feature Model and attach it to the root SPL node
Create AND/OR/XOR nodes in Feature Model	Create minimum of two feature nodes underneath the created node

5.5.2 Auto-Completion

Our MO not only helps to maintain model consistency but also performs a modeller's next edit when possible, either automatically or by offering a set of options to select from. This limited form of auto-completion is performed based on our consistency and semantic-correctness rules. For each auto-completion rule, we define an atomic *Head* action followed by one or more *Tail* actions. The Head action is a change to the model that is initiated by the modeller. The MO listens for Head actions; when one occurs, it automatically triggers one or more corresponding Tail action(s). For example, if the modeller moves an operation from a class to another class, the tool automatically updates all the navigation paths (e.g., guard expressions) that use the element's new navigation path expression. However, there must be a link (e.g., associations) between the classes in the Class diagram in order to update the navigation expression to the new navigation path. Otherwise, the tool will update the expression as the moved element preceded by its new parent class.

In Table 5.2, we present our rules. The Head actions are divided into three different groups: creating, deleting, and renaming model elements. In Fig. 5.2, we model the behaviours of our MO and auto-completion for each of the three types of Head actions (create, delete, rename) as a State-Machine diagram. For example, Fig. 5.2a models the cases that start with the modeller creating a model element. The MO listens to the model changes in the state *S0*. If there is a change that creates a model element, then it creates the element (*S1*) and automatically generates a unique name for it (*S2*). Then, depending on the type of the newly created element, there may be automatic Tail actions associated with it. For instance, if the newly created element is a State-Machine diagram, the MO automatically creates a main region (*S3*) with a unique name (*S4*), and an initial pseudo state for the main region (*S5*). Furthermore, it creates a new basic state in the region (*S6*) with a unique name for it (*S7*). Next, it creates a transition (*S8*) that connects the initial pseudo state and the new basic state (*S9*). Similarly, if the model change is a delete or a rename action (see Fig. 5.2b and Fig. 5.2c, respectively), the MO first checks whether the edited model element is used in any other diagram within the model. In case that it is used, the MO updates all the references, if possible. If there is only one possible update, the tool asks for the user's confirmation before proceeding with the update. If there are multiple possible updates, the tool asks the user to intervene by showing our Interactive Consistency Management interface.

The Head and Tail actions defined in Table 5.2 can significantly ease the modelling task by reducing the think time of the modeller (i.e., Z) as well as the time of the user's interaction with the tool (i.e., U). The automation provides several potential advantages (to be confirmed by empirical study), such as reduced workload and fatigue, avoiding (trivial) errors, machine utilization, increased precision in the performing of procedural tasks, and increased user productivity [109].

5.6 Focus+Context Interfaces

A **Focus+Context** interface is intended to allow the user to concurrently perform a *Focus* task while viewing and/or editing the relevant information to the task (i.e., *Context*) [21]. The main goal is to alleviate the user's navigation through several diagrams to locate specific information relevant to their *Focus* task, which help in reducing the modeller's memory workload. This section discusses the new Focus+Context interfaces that are proposed by us, which aim at alleviating the challenges of remembering contextual information when editing and debugging UML Class and State-Machine models.

Table 5.3 shows a list of focus elements or tasks, the diagrams in which they exist (denoted by $*$, † and ● icons), the contextual information relevant to the focus elements, and the types of actions that the user can perform on the contextual information. For instance, *FC1* refers to editing a transition expression, and the editor allows the user to view and modify the model elements in the Class diagram, as these model elements are among the vocabulary used to write precise transitions. Making this contextual information available to modellers while they are editing a transition expression helps modeller to recollect the elements that might be used in the transition expression without needing to switch back and forth among the different diagrams. Also, when the modeller is viewing and using the contextual information, it is the best time for them to detect errors or inadequacies in the contextual diagram. It is also easier to fix those errors on the fly without changing the editor's Focus. The focus actions and their corresponding contextual elements and actions are derived by 1) observing the subjects in our formative user study and the way they looked for and recollected contextual information, and 2) analysing FORML's meta-model to identify the meta-classes (*Context*) that can be referenced in other meta-classes (*Focus*).

We believe that the *Context*-based modelling challenge is mostly due to two difficulties: 1) *Recall*: consulting multiple relevant diagrams to find contextual information (e.g., a modeller may need to switch back to the Class diagram to recall an intended element that she wants to use in a state-transition expression in a State-Machine diagram), and 2) *Modify*: performing the prerequisite steps to some modelling task (e.g., defining a model element in the Class diagram before using it in a state-transition expression). In the following, we explain our Focus+Context interfaces starting with the most-important ones, namely the Focus+Context Transition Editor (related to *FC1*) and the Interactive

Table 5.3: List of focus elements with their contextual information.

ID	**Focus of User Action**	**Contextual Elements**	**Allowed Actions on Contextual Elements**
FC1*	Editing transition expressions	Elements in the Class diagram, Other Transitions in all of the State-Machine diagrams	Search, View, Modify, Reuse
FC2$^{* \dagger \bullet}$	Debugging an erroneous model	Elements relevant to the erroneous model element	View, Modify
FC3*	States and transitions in State-Machine	Elements in the corresponding Class diagram	Search, View, Modify
FC4*	States in State-Machine diagrams	Sub State-Machine (Fragmented State-Machine) diagrams	View, Open
FC5†	Editing classes in the Class diagram	Corresponding State-Machine diagram	View, Open
FC6†	Editing elements in the Class diagram	All the cross-references (i.e., usages) in the model	View, Modify
FC7†	Editing feature classes in the Class diagram	Feature Model (feature-tree)	View, Modify
FC8$^\bullet$	Editing feature nodes in the Feature Model	Corresponding State-Machine diagram, Relevant elements in the Class diagram	View, Open, Modify
FC9$^{* \dagger \bullet}$	Creating a new State-Machine diagram	Existing State-Machines	View, Reuse

$*$ Applicable to State-Machine Diagram
$\dagger$ Applicable to Feature Model Diagram
$\bullet$ Applicable to Class Diagram

Consistency Management Interface (related to *FC2*). Afterwards, we summarize other Focus+Context interfaces (related to *FC3* to *FC9*).

5.6.1 Focus+Context Transition Editor (FC1)

The **Focus+Context Transition Editor** is a novel techniques that is proposed by us to reduces the efforts of remembering and modifying the contextual information when editing a transition.

To design the Focus+Context Transition Editor we took a four-step design approach, which includes the following steps:

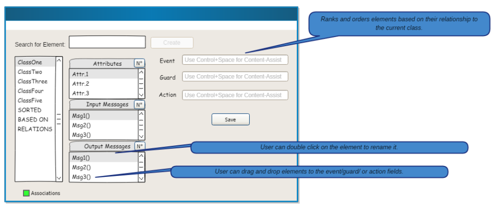
1. Sketch: We created eight possible individual design sketches of the interface (inspired by the Crazy 8's method [67]). Fig. 5.3 shows the eight sketches.

2. Present and prioritize: We then presented the sketches to five of the students (since five is usually accepted to be enough for usability testing [7]) who had modelling experience and have participated in our formative user study. We asked them to determine which sketches are more valuable then the others. Ultimately, they needed to determine one of the sketches as the most valuable design.

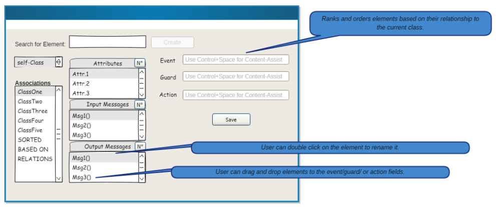
3. Critique and converge: We gave them a chance to offer feedback and critique the sketches by identifying the strengths and weaknesses of the sketches.

4. Choose and converge: We selected the sketch that gained more votes (i.e., Sketch No. 8 as shown in Fig. 5.3h) than the other sketches. We converged the feedbacks we received from the students on the strength and weaknesses of the sketches and enhanced the design accordingly. We iterated the above steps over the enhanced designs one more time to optimize the ultimate design.

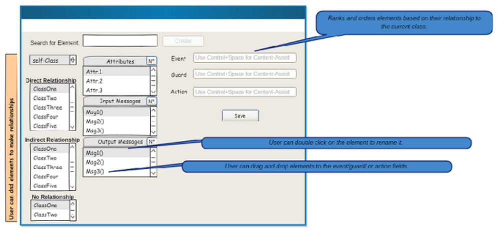
We implemented the **Focus+Context Transition Editor** (see Fig. 5.4) to alleviate the task of editing transition expressions. The Focus section (the upper right region of the editor window) pertains to the editing task; in this case, the task of writing a state-transition expression in a State-Machine diagram. Sections A, B, C, and D are used to alleviate the challenges of *Recall* and *Modify*. Table 5.4 lists all the user's possible contextual actions (i.e., *Recall* and *Modify* activities) related to the editing of a transition expression and shows which aspects of the editor's interface is designed to tackle each of the contextual actions (see the *Section* column). In the following, we introduce each section of the interface and explain how it can help alleviate the difficulties of *Recall* and *Modify*.


Search for Element:
Create
Ranks and orders elements based on their relationship to the current class.
ClassOne
ClassTwo
ClassThree
ClassFour
ClassFive
SORTED BASED ON RELATIONS
Attributes
Attr.1
Attr.2
Attr.3
Input Messages
Msg1()
Msg2()
Msg3()
Output Messages
Event
Guard
Action
Use Control+Space for Content-Assist
Save
User can double click on the element to rename it.
User can drag and drop elements to the event/guard/ or action fields.
Associations

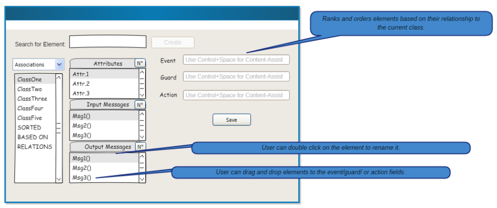
(a) Sketch No. 1.


Search for Element:
Create
Ranks and orders elements based on their relationship to the current class.
self-Class
Associations
ClassOne
ClassTwo
ClassThree
ClassFour
ClassFive
SORTED BASED ON RELATIONS
Attributes
Input Messages
Output Messages
Event
Guard
Action
Use Control+Space for Content-Assist
Save
User can double click on the element to rename it.
User can drag and drop elements to the event/guard/ or action fields.

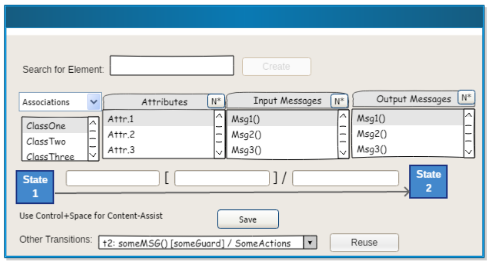
(b) Sketch No. 2.


User can dnd elements to make relationships
Search for Element:
Create
Ranks and orders elements based on their relationship to the current class.
self-Class
Direct Relationship
Indirect Relationship
No Relationship
Attributes
Input Messages
Output Messages
Event
Guard
Action
Save
User can double click on the element to rename it.
User can drag and drop elements to the event/guard/ or action fields.

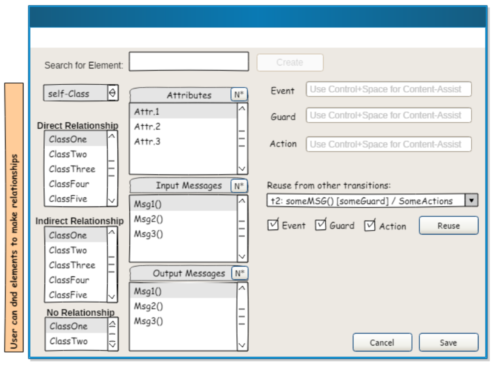
(c) Sketch No. 3.


Search for Element:
Create
Ranks and orders elements based on their relationship to the current class.
Associations
SORTED BASED ON RELATIONS
Attributes
Input Messages
Output Messages
Event
Guard
Action
Save
User can double click on the element to rename it.
User can drag and drop elements to the event/guard/ or action fields.

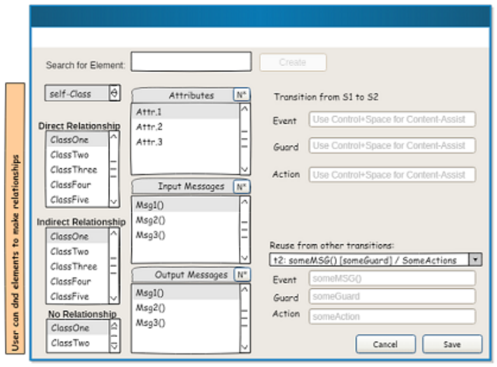
(d) Sketch No. 4.



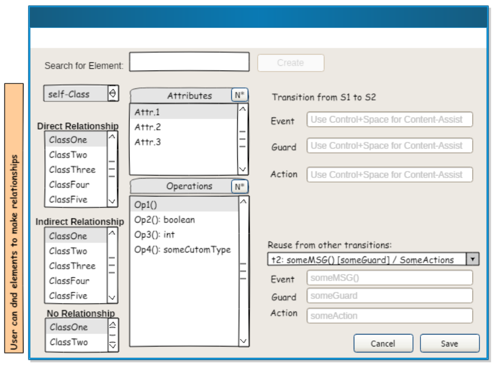
(e) Sketch No. 5.



(f) Sketch No. 6.

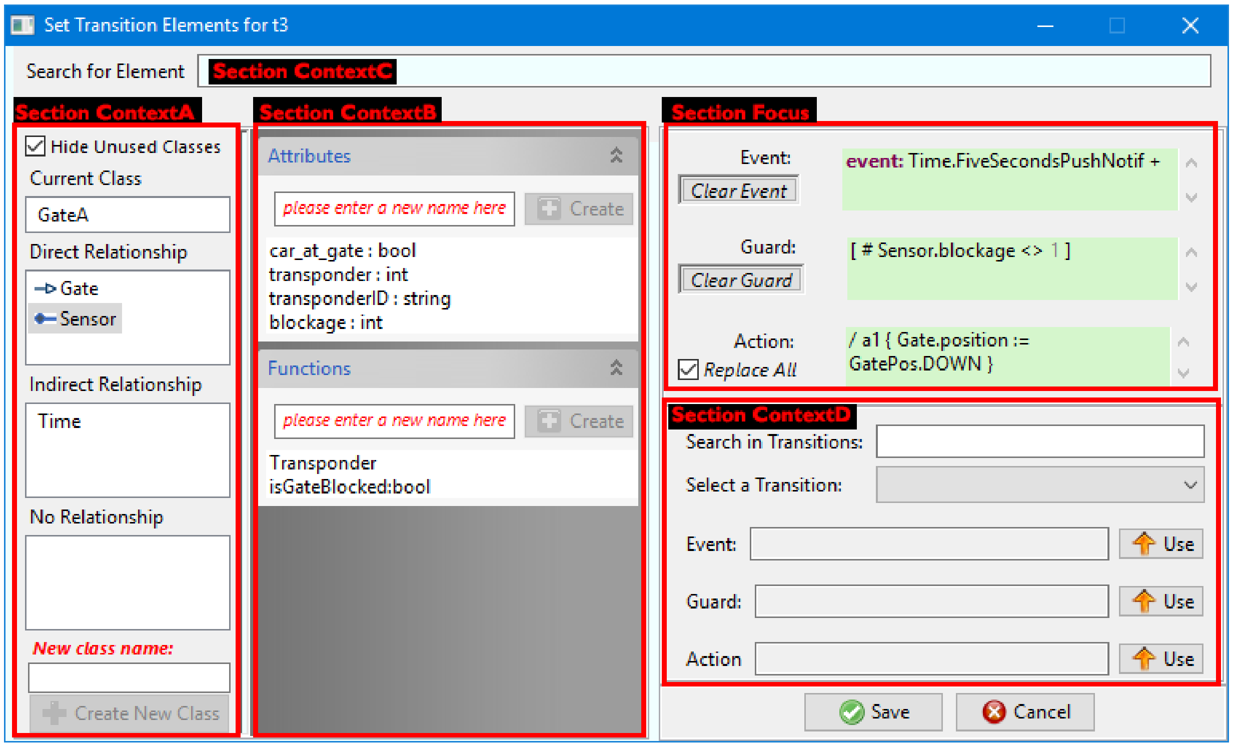


(g) Sketch No. 7.



(h) Sketch No. 8.

Figure 5.3: An overview of the eight sketches.



Current Class: GateA

Direct Relationship: Gate, Sensor

Indirect Relationship: Time

Attributes:
car_at_gate : bool
transponder : int
transponderID : string
blockage : int

Functions:
Transponder
isGateBlocked:bool

Event: **event:** Time.FiveSecondsPushNotif +

Guard: [# Sensor.blockage <> 1]

Action: / a1 { Gate.position := GatePos.DOWN }

Figure 5.4: Our Focus+Context Transition Editor.

- **Section ContextA:** A major difficulty that a user may face when editing a transition label is knowing about the classes, attributes, and operations declared in the Class diagram (*Rec.1*). It is also important to understand the relationships between the current class (whose State-Machine diagram is currently being edited) and other classes in the Class diagram (*Rec.2*). A modeller is more likely to refer to an attribute or operation of the current class or of an associated class, than to an attribute or operation of an unrelated class. The purpose of Section *ContextA* is to display the relationships among the classes based on the data collected from the Distance-Oriented Object Indexer (Section 5.2). Specifically, modellers are informed about the current class, the classes that have a *Direct Relationship* with the current class (the type of relationship is illustrated by an icon), the classes that have an *Indirect Relationship* (i.e., connecting through another class), and the classes that have *No Relationship* with the current class. This categorisation of the classes allows the modeller to focus on those classes to which they will most likely need to refer.

  If a user needs to create an association between the classes (*Mod.2*), he/she can drag

Table 5.4: List of Recall and Modify actions, and the relevant sections of the Focus+Context Transition Editor.

ID	Type	Description	Section(s)
Rec.1	Recall	Remembering the elements in the Class diagram (i.e., Classes, Attributes, Operations, Types)	A,B,C
Rec.2	Recall	Remembering the relationships amongst the entities in the Class diagram	A
Rec.3	Recall	Remembering/reusing expressions (i.e., event, guard, actions) that are used in other transitions	D
Rec.4	Recall	Looking for an intended element based on the modeller's recollection of its name	C
Mod.1	Modify	Creating an intended element (e.g., classes, attributes, operations) in the Class diagram	A,B,C
Mod.2	Modify	Creating or modifying a relationship between entities in the Class diagram	A
Mod.3	Modify	Changing the type of an element (e.g., attribute) in the Class diagram	B

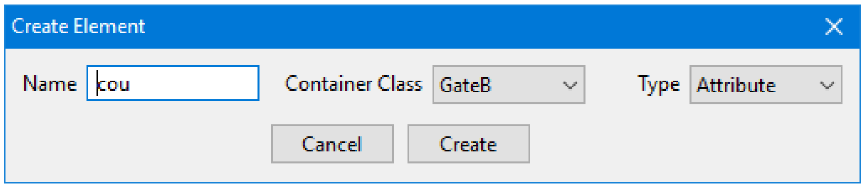
the class from the list of *Indirect Relationship* or *No Relationship* classes and drop it into the list of *Direct Relationship* classes. The editor will, by default, create a direct *association* between the current class and the dragged class. The user can then right click on the newly associated class and modify the type of the relationship from association to composition, aggregation, or generalization. As another example, the user can create a new class (*Mod.1*) by typing the name of the class in the text field that is located at the bottom of the section. The editor will, by default, insert the newly created class in the *No Relationship* category, but the user can use drag-n-drop to move it to the *Direct Relationship* classes.

Lastly, using *"Hide Unused Classes"* check box, the user can view only those classes whose elements have been previously used in the current State-Machine. The hypothesis behind this feature is that modellers are more likely to use mostly the elements of a subset of classes when developing a State-Machine. Thus, if a user needs to update a State-Machine, she may not need to view all the classes that exist in the Class diagram.

- **Section ContextB:** A modeller of a state-transition expression may continuously refer to the Class diagram to look for the attributes and operations of classes (*Rec.1*) as these elements are the most-likely candidates to be referenced in a transition label.

Section *ContextB* lists the attributes and operations from the class selected in Section *ContextA*. In Fig. 5.4, *ContextB* lists the attributes and operations of the *Sensor* class selected in *ContextA*. The user can then drag an attribute or an operation from *ContextB* and drop it into one of the text fields in Section *Focus*. If the intended attribute or operation does not exist in the selected class, the user can use the *Create* buttons to create them (*Mod.1*). The user can right-click on an attribute or operation to change its type (*Mod.3*).

- **Section ContextC:** To help users overcome the challenges of remembering an intended element, Section *ContextC* enables the modeller to look for an element based on what they remember of the element's name. As the user types into the search text field, the Distance-Based Model Element Finder (Section 5.4) displays in real-time the list of elements in the model whose names most-closely match the typed name. It first lists all elements whose names match exactly the typed word (*Rec.1*). If there are more than one exact-matches, it orders them based on their distance to the current class. If none of the exact-matches or similar-matches are the element the modeller is looking for, then the modeller is invited to create a new element (only class attributes or operations) with the name of the typed word (*Mod.1*). For this, the tool pops up a new dialogue box (see Fig. 5.5) asking the user to confirm the new element's name, the containing class for the new element, and determine whether the new element is an attribute or an operation.


Create Element

Name: cou

Container Class: GateB

Type: Attribute

Figure 5.5: Creation of an attribute or operation using Section ContextC.

Fig. 5.6 shows an example where the user types *"cou"*. As can be seen in the figure, the first item invites the user to create the *"cou"* element (which displays Fig. 5.5, if selected), and the second and the third elements refer to the *gateCounter* attribute (of class *Gate*) and *increaseCounter* operation (of class *Parking*), which contain *"cou"* as part of their names. Furthermore, the editor ranked the *gateCounter* attribute before the *increaseCounter* operation because the distance between the current class (for which we are developing State-Machine) and the *Gate* class is less than the distance between the current class and the *Parking* class.

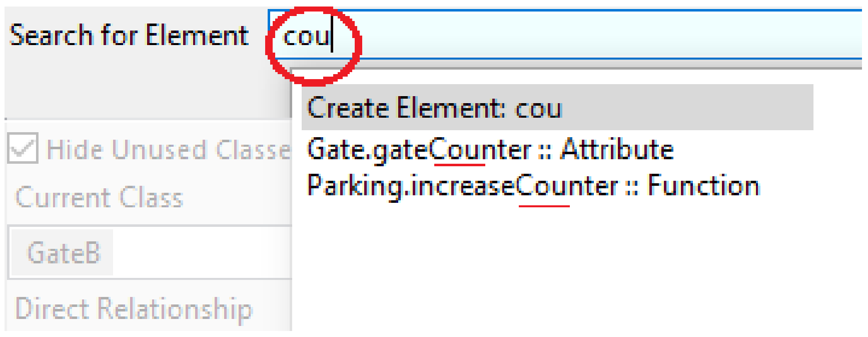


Figure 5.6: An example list of items as the result of user's search.

- **Section ContextD:** The results of our prior user studies suggest that users often look for a similar transition to reuse or learn from when editing another transition's expression [80]. This, however, can be cumbersome as it may not be easy to find a specific transition in a large and complex model. In Section *ContextD*, we allow the modeller to search among the State-Machine diagrams in the model for other transitions based on their name, or elements in their labels (*Rec.3*). As the user types in the *ContextD* search field, the editor continuously filters and lists all matching transitions whose expression (label, event, guard, and actions) includes an exact-match or similar-match to the typed expression. Results are listed by showing their names and sub-expressions in the drop-down menu that is labelled *Select a Transition*. Subsequently, when a transition is selected from the drop-down menu, its event, guard, and action segments are inserted into the corresponding text fields in Section *ContextD*, and the user can click on any of the *Use* buttons to reuse one of the sub-expressions in Section *Focus*.

- **Section Focus:** This section provides aids that improve the modeller's *Focus* on the current editing task. It divides the task into three text fields which correspond to the event, guard, and action segments of the transition expression that is being edited and it supports two different ways of setting these segments: *Feature-Rich Text Fields* and *Drag-n-Drop*.

  Feature-Rich Text Fields provide various capabilities to the users such as syntax-highlighting, displaying errors and warnings, and Content-Assist that together nudge the modeller towards a correct model and warn the modeller when they violate a consistency rule. Warnings and errors notifications are issued in realtime as rule violations are committed. In contrast, the Content-Assist is user-activated and uses *Type-Based and Distance-Oriented Ranking* module to list related model elements in order of hypothesized relevance.

Drag-n-Drop: By using Drag-n-Drop, users can drag an intended operation or attribute from Section *ContextB* and drop it into the event, guard or action text fields of the *Focus* section. When editing the guard expression, it is possible that the user wants to write a conditional expression (e.g., $A==B$) of which a dropped attribute or operation is one operand. When the modeller drops one operand into the text field, the editor will automatically create a conditional expression and assign the dragged element to the LHS of the conditional expression. The editor then uses Distance-Oriented Object Indexer (Section 5.2) to provide a prioritized list of possible elements and values that can be assigned to the RHS of the conditional expression based on the type of the dropped element (i.e., LHS). Moreover, the equality check (==) is inserted as the default conditional operator but can be changed by the user. The editor analogously assists the user when they drop an attribute or operation into the action text field, to complete an assignment statement.

Given a *Drag-n-Drop* action, the editor, by default, will replace the old expression with a new expression. Thus, if the user drops an operation into the event field, the new triggering event will be automatically set to the dropped operation as there can be only one operation assigned to an event. However, unlike an event, a guard expression can be composed of multiple conditional statements (e.g., $A==B$ OR $C==D$); and an action expression can be composed of multiple actions. Therefore, the editor supports a keyboard shortcut that cause a *dropped* element to extend rather than replace a guard or action. By default the editor extends a guard expression with an *AND* operand and then with a new conditional sub-expression whose first operand is the dropped element. The user can replace the *AND* operand with an *OR* or *XOR* operand. The editor only supports an *AND* operand between existing action expressions and a newly dropped element that forms the LHS of a new assignment action. For example, Fig. 5.7 shows that the user drags the *pos* attribute (of the *Gate* class) and drops it into the transition's action text box. The editor pops up the Content-Assist once the user drops the element into the text box, and because the *pos* attribute is from the *GatePos* enumeration type, the Content-Assist only contains the constants of the *GatePos* enumeration class (which are *DOWN* and *UP*) and other tokens that are valid based on the language's grammar.

We hypothesize that: *Providing editing facilitators for the* ***Recall*** *and* ***Modify*** *activities of modelling can reduce the efforts related to a* ***Focused*** *modelling task (i.e., editing a transition).*

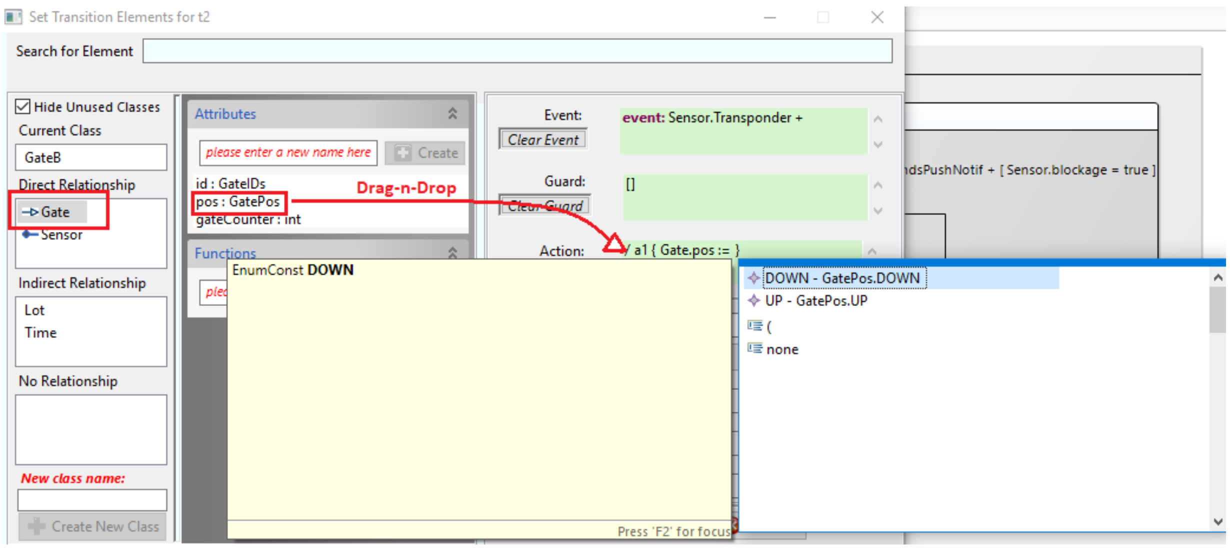


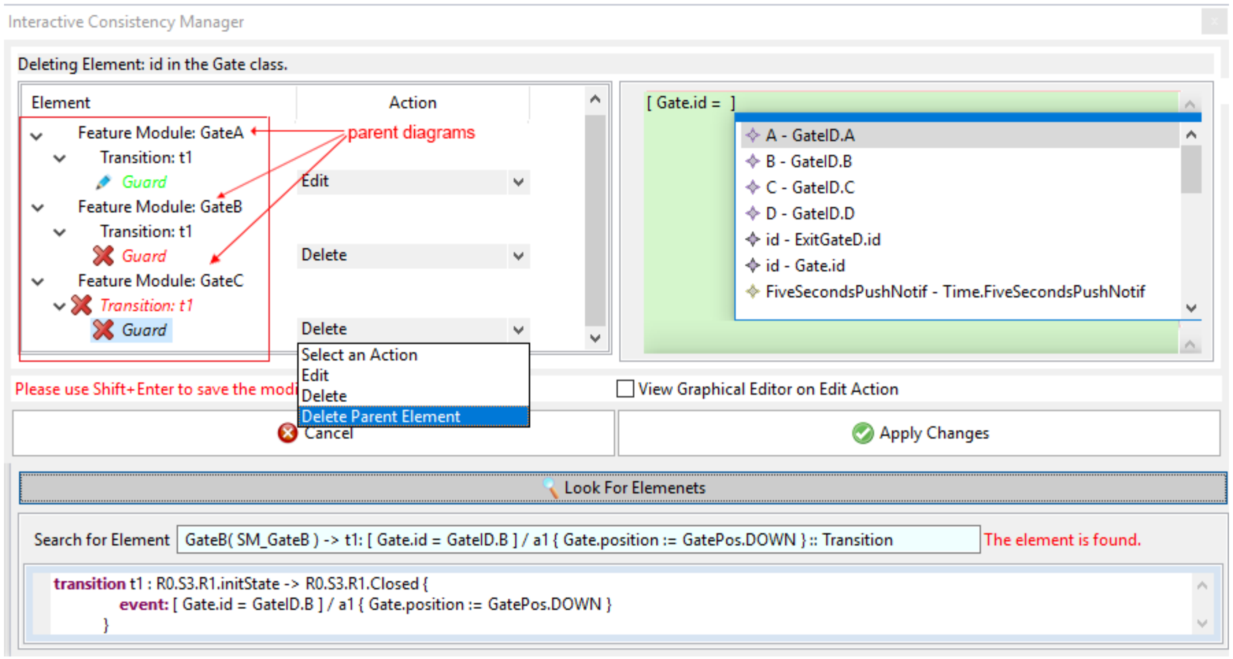
Figure 5.7: An example of Drag-n-Drop in the transition editor.

5.6.2 Interactive Consistency Management Interface (FC2)

In this subsection, we propose a new interface, namely **Interactive Consistency Management Interface**, which aims at facilitating the modeller's task of debugging models. In addition to reducing the modeller's memory workload, it helps in improving decision-making and action planning abilities of modellers, avoiding rule-based and transience-related errors, and reducing errors due to absentmindedness, lack of context, and lapses.

A typical large-scale system may be composed of hundreds of classes and corresponding State-Machine diagrams [47][56]. Maintaining consistency in such large-scale systems requires a lot of effort due to the number of inter-related elements among the different diagrams. In our opinion, for a consistency management approach to be successful, it should take into account the following user-centric concerns: Recognizing the sources of inconsistency, alerting the user of an inconsistency without being intrusive, ensuring that the user can easily access all relevant information (*Context*) needed to resolve the inconsistency, and supporting the users in their plans to resolve an inconsistency.

Recognition of Errors: The first step to resolve an inconsistency is to recognize when a model edit has introduced an inconsistency in the model. The recognition process can be influenced by two memory-related factors: 1) *Transience:* a measure of how easily information can be retrieved from the modeller's memory and the degree to which memory deteriorates over time [93]; and 2) *Absentmindedness*: lapses in paying attention [93]. A modeller can fail to recognize an occurred inconsistency, or they recognize the inconsistency but they decide to resolve the inconsistency at a later time but they forget. To avoid such failures to address inconsistencies, our tool attempts to locate and resolve inconsistencies while they are being made.



Deleting Element: id in the Gate class.

[Gate.id =]

Please use Shift+Enter to save the modi

GateB(SM_GateB) -> t1: [Gate.id = GateID.B] / a1 { Gate.position := GatePos.DOWN } :: Transition

The element is found.

```
transition t1 : R0.S3.R1.initState -> R0.S3.R1.Closed {
        event: [ Gate.id = GateID.B ] / a1 { Gate.position := GatePos.DOWN }
    }
```

Figure 5.8: Our Interactive Consistency Management interface.

We hypothesize that it is easier and faster to recognize and correct errors while they are being made because the error-inducing part of the model is still fresh in the modeller's mind [27]. Our editor includes three error detection and resolution strategies: Auto-Fix, Quick-Fix, and Interactive-Fix, all of which attract the user's attention to a new error. Auto-Fix and Quick-Fix approaches were described in Section 5.5. Our Interactive Consistency Management Interface (also referred to as Interactive-Fix interface), shown in Fig. 5.8, pops up a dialogue box during model-editing whenever a detected error is complex enough to require greater user input.

Intrusiveness: It is often argued that resolving inconsistencies on-the-fly during modelling is disruptive and counter-productive to users because of the memory overload imposed by the context switching among the different diagrams to locate an error [27][39]; and because the interruption distracts the user from their original train of thought [3][8][9][13]. In contrast, we hypothesize that recovering errors in real-time need not adversely affect a user's performance and satisfaction. We devise a model editor that mitigates the disruptive impacts of interruptions while enabling on-the-fly error resolution.

Make the Invisible Visible: One of the cognitive factors involved in the process of maintaining model consistency is *Association*, which refers to cross-linking a piece of information to related pieces of information. That is, the ability to remember and list the cross-linked and out-of-sight (or out-of-mind) model elements that may become inconsistent as a result of an edit. Relying on a user's ability to locate and co-modify the associated elements may lead to failure and model inconsistencies as the user may forget to check (i.e., lapses) or miss (i.e., slips) some of the affected locations in the model.

To promote the *Association* (cognitive) ability, our User-Interactive-Fix interface (see Fig. 5.8) provides a hierarchical view (bordered in red color in the figure) of all of the inconsistent elements in the model that are affected by an original edit. The root node in the hierarchy refers to the diagram that contains the erroneous elements (i.e., the elements that become inconsistent as the result of a model edit). For example, Fig. 5.8 shows that the *id* attribute of the *Gate* class has been deleted from the Class diagram, which introduces errors in different guard expressions in the model. Accordingly, the root node is the State-Machine diagram that contains the transition (parent node) of the erroneous guards (leaf nodes) -e.g., guard expression of transition *t1* in the *FeatureModule: GateB* diagram.

Action Planning: An important step in problem-solving and decision-making is to choose from the set of possible alternative actions. Most editors do not assist the modeller in fixing an inconsistency when it occurs; a few editors suggest that the modeller undo the edit or delete the inconsistent element (e.g., Capella [89]). Our tool allows the modeller to select (using an **Actions-To-Do** menu) editing or deleting the newly inconsistent element, or its parent element. For example, if a newly inconsistent element is a guard, then the user can select to edit the guard expression (using the embedded textual editor), delete the guard element, or delete the parent transition. Note that the edit action is not limited to the guard expression: the user can easily access and edit the event or actions of the parent transition by clicking on the parent node in the hierarchical view.

Context: Recall that *Context* [17] is another critical cognitive factor influencing the process of resolving inconsistencies. It refers to remembering and integrating the essential pieces of related information that the user needs to edit (or delete) an inconsistent element. This information may be spread across other diagrams than the *Focus* diagram, thus requiring the user to switch amongst different diagrams, which can be cumbersome. Insufficient access to and recall of a task's *Context* may result in making Knowledge-Based mistakes.

To support the *Context* ability, our User-Interactive Interface embeds a *Feature-Rich Textual Editor* (the green-colored text box located at the right side of the interface in Fig. 5.8) which presents the textual representation of a selected inconsistent element and allows the modeller to edit the element. The textual editor uses our *Type-Based and Distance-Oriented Content-Assist* module to facilitate the task of editing by proposing and prioritizing relevant model elements. As an alternative, our tool provides a graphical editor that allows the modeller to view and edit the containing diagram (automatically navigates to the diagram segment that contains the element).

In summary, we hypothesize that: *Our User-Interactive Consistency Management interface improves the users' effectiveness in creating more correct models and reduces users' feelings of being disrupted when fixing an inconsistency by taking into account the relevant human-cognition factors.*

5.6.3 Other Focus+Context Interfaces

Table 5.3 lists nine Focus+Context interfaces that we implemented in our tool. The above two subsections described the important interfaces (i.e., *FC1* and *FC2*). The following presents the rest of the new interfaces that we propose to alleviate the modeller's challenges of remembering the contextual information while performing several modelling tasks. Similar to other Focus+Context interfaces, the following interfaces mainly decrease the modeller's memory workload.

Class Elements Viewer (FC3). Whenever the user selects an element (i.e., a state, or a transition) in a State-Machine diagram, they should be able to view and modify the corresponding classes. For example, the Focus+Context Transition Editor, the context of a transition expression includes the classes whose elements (i.e., attributes or operation) are used in the transition's event, guard expression, or actions, plus the class for which the current State-Machine is being edited. When selecting a state, the *Context* consists of all the classes that are used in the state's incoming and outgoing transitions. Fig. 5.9 shows an example of the Class Elements Viewer. As shown in the figure, the user has selected the *Closed* state (*Focus*), which has one incoming transition and one outgoing transition: *t1* and *t2*. As the *Context*, the tool lists *GateC*, *Gate*, and *Sensor* classes as well as their attributes and operations in the bottom view (*Context*), which correspond to the class for the current State-Machine and the classes whose elements are used in *t1* and *t2*, respectively. Moreover, the user can create new attributes or operations for each of the classes. Additionally, if the user needs to view another class and its elements that is not listed in the bottom view, she can use the *"Search for Element"* text field to look

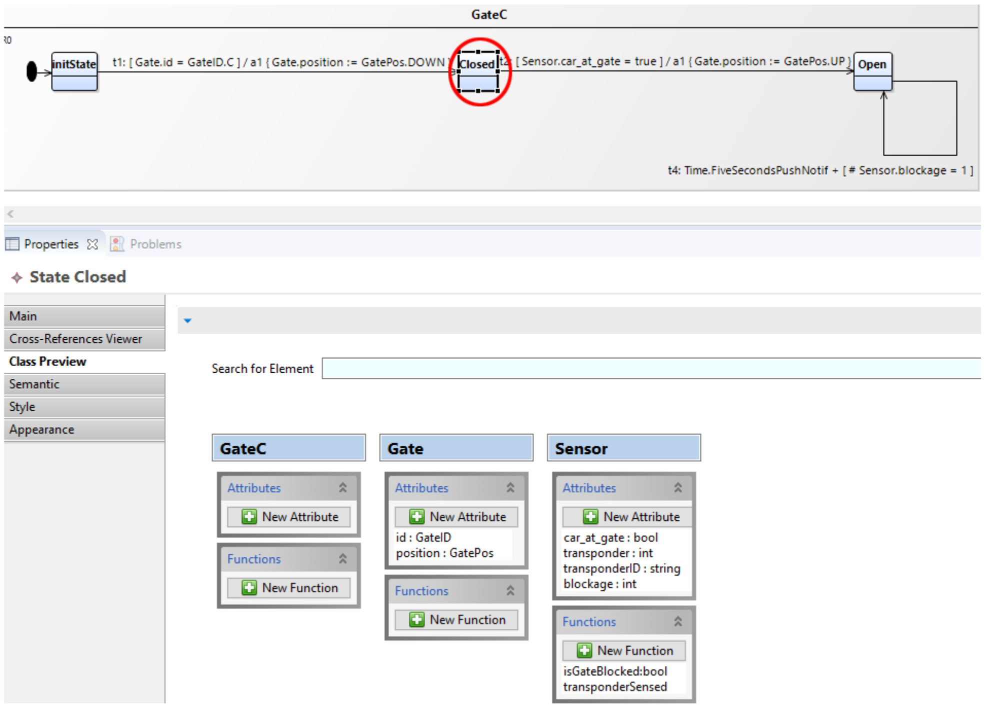


Figure 5.9: An example view of the Class Elements Viewer.

for the desired class and add it to the list of classes in the bottom view. The editor uses our Distance-Based Model Element Finder to suggest the classes based on their names' similarity with what has been typed by the user (i.e., a modeller can find a class simply by typing what they think is the class name). Moreover, they can also search for a class by typing the name of the class's attributes or operations in the text field. Thus, the user does not need to remember the exact name of the element that they are looking for. We believe such an interface can minimize the user's switches between State-Machine and Class diagrams.

Fragmented State-Machine Viewer (FC4). As mentioned in 2.2.3, a modeller can create sub- (or fragmented) State-Machine from an existing State-Machine to reuse its behaviour. When the user selects a state (s) in a State-Machine (SMp), the tool previews the fragment (sub-) State-Machines (SMf) of the selected state (s), if any. Fig. 5.10 shows an example in which the tool previews a fragment State-Machine of the *disengaged* state in the *TokenPay* State-Machine. If the user opens a State-Machine (SMf) that is fragmented from a state (s) in another State-Machine (SMp), the tool previews the parent State-Machine (SMp) in the bottom view.

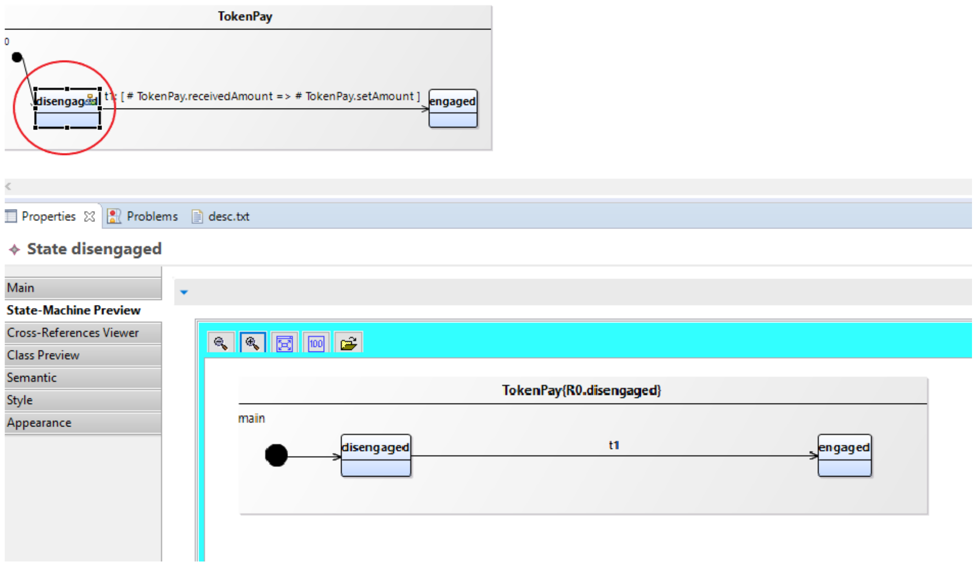


Figure 5.10: An example view of the Fragmented State-Machine Viewer.

One can notice that, there are more than one contextual interfaces when selecting a state, namely the Class Elements Viewer (*FC3*) and Fragmented State-Machine Viewer (*FC4*). In this case, the tool lists all of the contextual interfaces and the user can select the desired one. For example, in Fig. 5.10, the tool lists both *State-Machine Preview* and the *Class Preview* in the left-side panel of the *properties* view which refer to *FC3* and *FC4*, respectively.

State-Machine Viewer (FC5). By selecting a class in the Class diagram, the user can view its corresponding State-Machine diagram (see Fig. 5.11), if any. This helps the user gain information about the dynamic aspect of an entity without switching to the entity's respective diagram. Moreover, the State-Machine Viewer also provides mechanisms to open the diagram in a separate window if the user needs to edit the State-Machine in its own window.

Cross-Reference Viewer (FC6). By selecting any element (i.e., attribute, operation, or class) in the Class diagram, the tool shows a list of all the places where the selected element is used. Fig. 5.12 shows an example of the Cross-Reference Viewer, where the user has selected the *blockage* attribute of the *Sensor* class (*Focus*). In response, the tool lists in the bottom view all of the places in the model where the *blockage* attribute is used (*Context*). By selecting each of the references, the user can view a textual representation of the selected reference and can edit its value using the Feature-Rich Textual Editor without

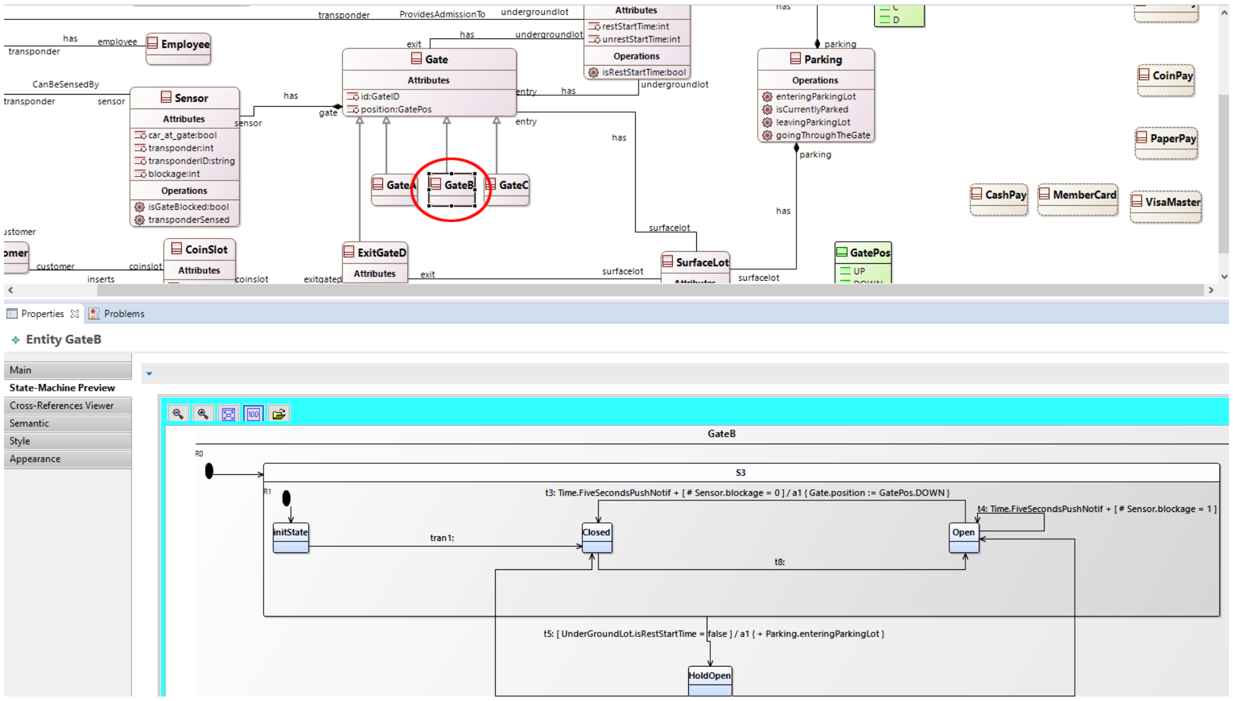
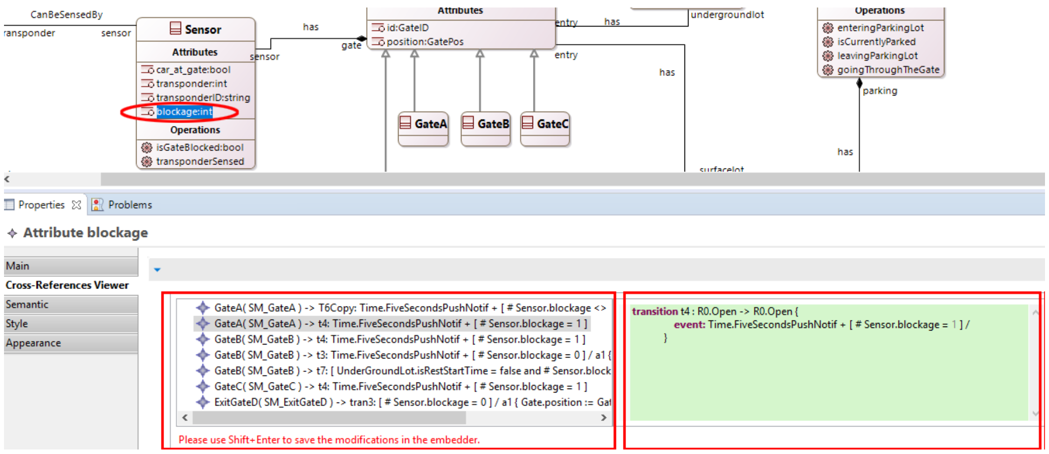

Properties
Problems
Entity GateB
Main
State-Machine Preview
Cross-References Viewer
Semantic
Style
Appearance
GateB
R0
S3
R1
t3: Time.FiveSecondsPushNotif + [# Sensor.blockage = 0] / a1 { Gate.position := GatePos.DOWN }
t4: Time.FiveSecondsPushNotif + [# Sensor.blockage = 1]
initState
tran1:
Closed
Open
t8:
t5: [UnderGroundLot.isRestStartTime = false] / a1 { + Parking.enteringParkingLot }
HoldOpen

Figure 5.11: An example view of the State-Machine Viewer.



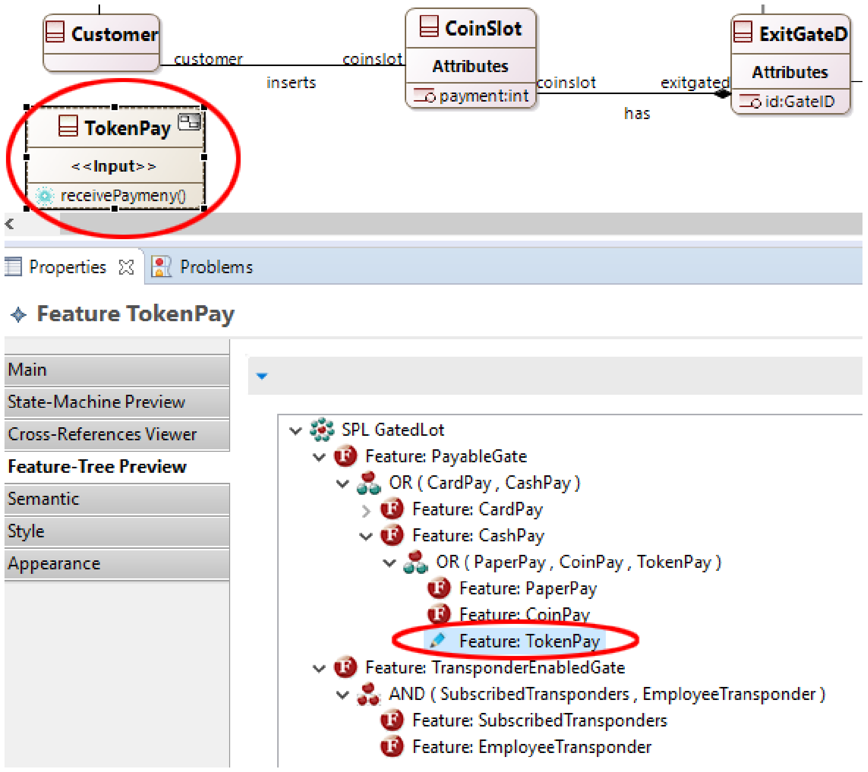
```
GateA( SM_GateA ) -> T6Copy: Time.FiveSecondsPushNotif + [ # Sensor.blockage <>
GateA( SM_GateA ) -> t4: Time.FiveSecondsPushNotif + [ # Sensor.blockage = 1 ]
GateB( SM_GateB ) -> t4: Time.FiveSecondsPushNotif + [ # Sensor.blockage = 1 ]
GateB( SM_GateB ) -> t3: Time.FiveSecondsPushNotif + [ # Sensor.blockage = 0 ] / a1 {
GateB( SM_GateB ) -> t7: [ UnderGroundLot.isRestStartTime = false and # Sensor.block
GateC( SM_GateC ) -> t4: Time.FiveSecondsPushNotif + [ # Sensor.blockage = 1 ]
ExitGateD( SM_ExitGateD ) -> tran3: [ # Sensor.blockage = 0 ] / a1 { Gate.position := Gat
Please use Shift+Enter to save the modifications in the embedder.
```

```
transition t4 : R0.Open -> R0.Open {
        event: Time.FiveSecondsPushNotif + [ # Sensor.blockage = 1 ] /
    }
```

Figure 5.12: An example view of the Cross-Reference Viewer.

opening the referring diagram.

Feature Model Viewer (FC7). When the user selects a feature class in the Class

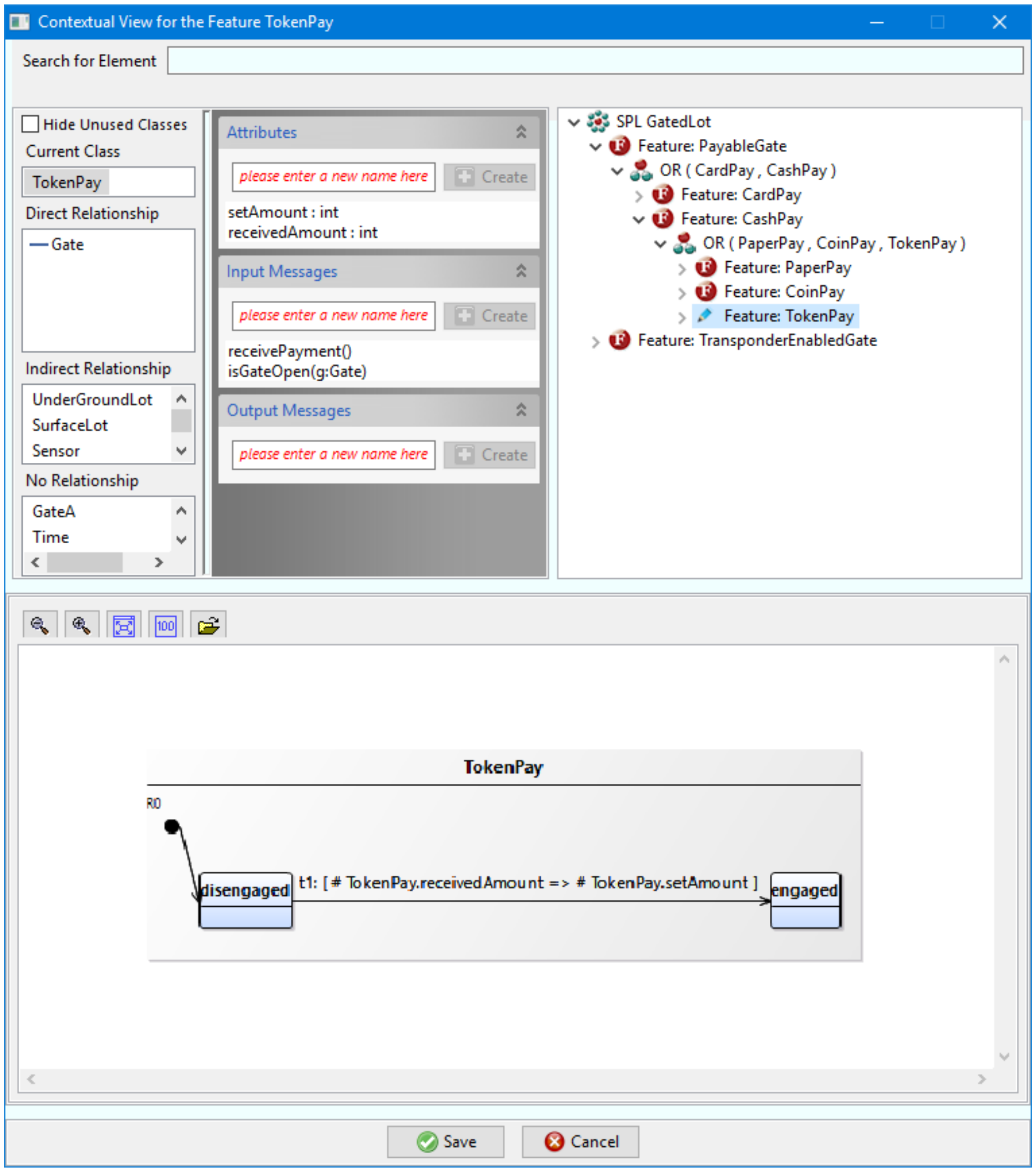


```
SPL GatedLot
  Feature: PayableGate
    OR ( CardPay , CashPay )
      Feature: CardPay
      Feature: CashPay
        OR ( PaperPay , CoinPay , TokenPay )
          Feature: PaperPay
          Feature: CoinPay
          Feature: TokenPay
  Feature: TransponderEnabledGate
    AND ( SubscribedTransponders , EmployeeTransponder )
      Feature: SubscribedTransponders
      Feature: EmployeeTransponder
```

Figure 5.13: An example view of the Feature Model Viewer.

diagram, the editor displays its corresponding feature node in the SPL's feature tree (i.e., Feature Model) within the Feature Model Viewer (see Fig. 5.13). The user can also modify (e.g., reorganize) the features in the feature-tree by dragging and dropping the nodes. Also, the modeller can edit the type of the feature OR, AND, and XOR nodes. For example, the modeller can right-click on an OR node and change it to an AND node, provided that the constraints are met (i.e., there should be minimum of two children feature nodes for an AND node).

Feature Class Viewer (FC8). One of the sources of contextual information related to editing a Feature Model is the information about the classes and their properties in the Class diagram. We designed a Focus+Context interface to bring such contextual information into the user's view when editing the Feature Model. That is, the user can double click on a feature node in the Feature Model to view the interface shown in Fig. 5.14. On the right-side panel of the interface, the user can re-organize the feature tree (e.g., moving features around). When the user selects a feature from the right-side panel,


Contextual View for the Feature TokenPay

Search for Element

Hide Unused Classes

Current Class

TokenPay

Direct Relationship

Gate

Indirect Relationship

UnderGroundLot
SurfaceLot
Sensor

No Relationship

GateA
Time

Attributes

setAmount : int
receivedAmount : int

Input Messages

receivePayment()
isGateOpen(g:Gate)

Output Messages

SPL GatedLot
- Feature: PayableGate
  - OR (CardPay , CashPay)
    - Feature: CardPay
    - Feature: CashPay
      - OR (PaperPay , CoinPay , TokenPay)
        - Feature: PaperPay
        - Feature: CoinPay
        - Feature: TokenPay
- Feature: TransponderEnabledGate

TokenPay

R0

disengaged

t1: [# TokenPay.receivedAmount => # TokenPay.setAmount]

engaged

Figure 5.14: An example view of the Feature Class Viewer.

the interface displays its corresponding Feature Module in the bottom-side panel. Similar to the Focus+Context Transition Editor, the user can also view and modify the elements of the Class diagram using the left and center sections of the interface.

State-Machine Duplicator (FC9). Sometimes, the user prefers to reuse an existing

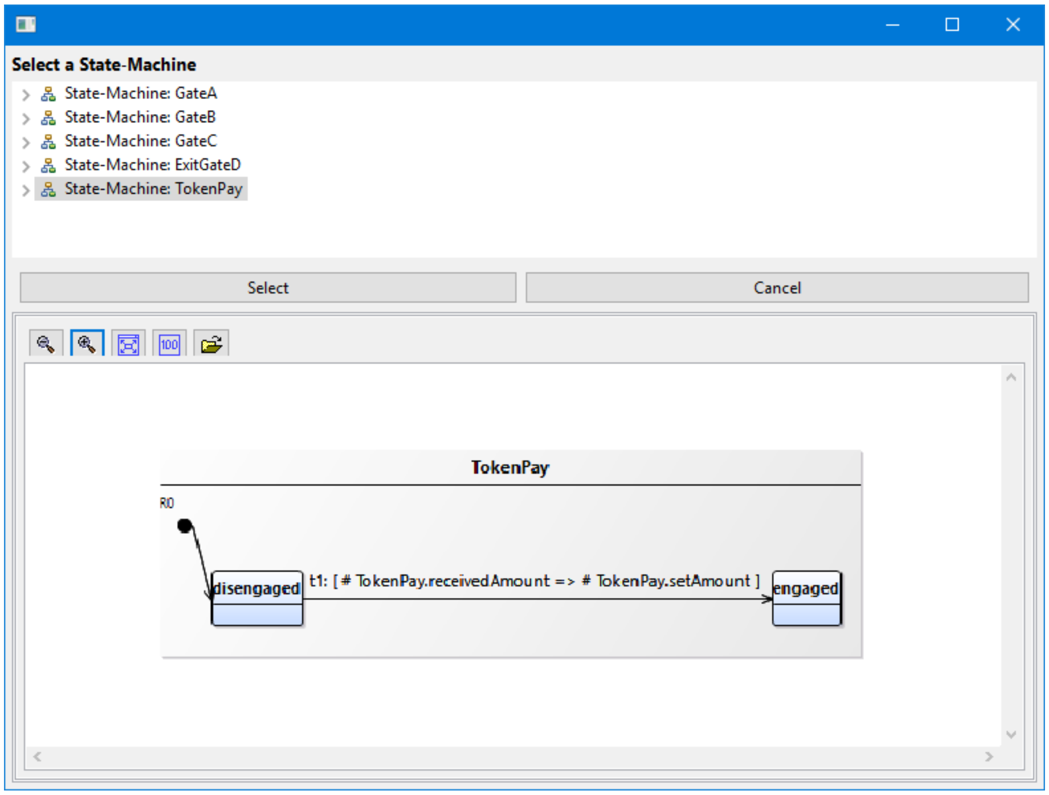


Figure 5.15: An example view of the State-Machine Duplicator.

State-Machine diagram when defining a new one because the two State-Machines have similar structures. As shown in Fig. 5.15, our tool allows the user to create a new State-Machine based on an existing State-Machine. The figure shows that the user can view a list of available State-Machines and preview each one of them before moving forward with reusing it.

The Focus+Context Interfaces described above can alleviate the modelling tasks by reducing the think time of the modeller (i.e., Z) as well as the time of the user's interaction with the tool (i.e., U) as the user does not need to back and forth amongst different diagrams to recollected the contextual information. The interfaces can introduces potential benefits to the modellers, such as reducing memory workload and enhancing the user's experience of creating more correct an precise models.

Table 5.5: Summary of tooling enhancements and their relevant cognitive theories.

Feature	**Involved Cognitive Theory**
Auto-completion	Decreases execution and planning-related failures.
Contextual information in Focus+Context interfaces	Decreases short-term and long-term memory workload.
Contextual information (in Focus+Context interfaces or Content-Assist)	Decreases short-term and long-term memory; decreases number of errors due to lapses or knowledge-based mistakes; less misattribution.
Distance-based search boxes	Reduces short-term and long-term memory workload.
On-the-fly error resolution	Avoids rule-based and transience-related errors.
List of inconsistencies in a single interactive view	Reduces errors due to absentmindedness, lack of context, and lapses.
Feature-rich textual embedded editor	Reduces memory workload and execution errors.
Action-to-do list in the Interactive Consistency Management interface	Improves decision-making and action-planning.

5.7 Concluding Remarks

In this chapter, we described our tool which facilitates the development of Feature-Oriented and UML-like models by employing and balancing user-centric and artefact-centric techniques. The rationale behind our tool is to consider the human in the loop when designing the enhancements rather than taking into account only the language specifications (e.g., UML's specification). In particular, our goal is to alleviate the cognitive challenges of modellers when remembering the contextual information that is relevant to performing a particular task. Furthermore, our tool generally aims to reduce the cognitive efforts to develop models of formal languages by applying semi-automated repairs and auto-completions in the course of (formal) modelling. Table 5.5 lists a summary of our tool's features and the involved relevant cognitive theories.

Some of our tooling advances are borrowed and adapted from existing tools, and some are invented:

- Focus+Context Transition Editor (FC1): We *introduce* a novel Focus+Context Transition Editor that displays all the contextual information in an integrated interface, which helps users to avoid switching back and forth among different diagrams. In addition, users can modify the contextual information (e.g., adding new class elements) if needed. The interface also allows the user to set different parts of a transition by reusing parts of an already developed transition.

- Interactive Consistency Management (FC2): In our consistency management method, we *borrowed* the Auto-Fix feature, which can be found in other tools (e.g., [34][70][89]). We *extended* Eclipse's Quick-Fix feature to provide users with Quick-Fixes (i.e., recommendations) on how to address most of the common errors (based on [54]) that can occur when editing models. We *introduce* the User-Interactive Consistency Management interface, which is a novel feature to help modellers with the task of resolving complicated inconsistencies. Using XText [26] to implement our formal language [97] enabled us to embed our customized Content-Assist [94] in our tool. We redesigned the default Content-Assist to employ a distance-based method for ordering and ranking the model elements to be presented to users when developing transition labels. Additionally, our Content-Assist embeds a novel filtering mechanism which takes a proactive approach and filters out those model elements that are not sound by type. The only tool that we are aware of having a similar type-checking mechanism is Yakindu [70], which takes a reactive approach and allows the modeller to write an expression and then throws an error if the two sides of a condition or an assignment are not from the same type.

- Other Focus+Context Interfaces (FC3 to FC9): None of these interfaces are naturally supported in the underlying framework that we use. These interfaces are *introduced* by us based on the observation of the users working with the UML-like model editors and the analysis of the meta-model of the UML and the FORML.

We believe that the proposed techniques provide insight for tool vendors into how to improve the usability and utility of model editors, by employing a human-centric approach that reduces users' cognitive challenges.

Chapter 6

Empirical Evaluation

"Computer scientists make broad claims for the simplicity, naturalness, or ease-of-use of new computer languages or techniques, but do not take advantage of the opportunity for experimental confirmation."

— – Ben Shneiderman, *"Software Psychology"[98]*

This chapter[7] presents the results of our empirical evaluation of our tooling techniques. We designed two user studies (a Context study and a Debugging study) to evaluate the effectiveness of our tooling advancements. We designed the experiments according to the guidelines from Tullis and Albert [7] and Pietron et al. [78], with respect to the number of subjects and data-collection techniques.

6.1 Experimental Design

In this section, we describe the design of these experiments in detail (see Appendix D for the materials of the user studies.).

[7]This chapter is reprinted in adopted form from [81].

6.1.1 Goals

The goals of the Context and Debugging user studies are explained using the Goal-Question-Metric (GQM) template in Table 6.1 and Table 6.2, respectively.

Analyze the Focus+Context Transition Editor (as an example of our Focus+Context editors)
For the purpose of understanding its effectiveness in alleviating the challenges in remembering contextual information
With respect to the users' performance, correctness, efficiency, frequency of experienced Context-related challenges, and satisfaction
From the perspective of software modellers
In the context of model-editors for UML Class and State-Machine diagram modelling.

Table 6.1: Goal of the Context user study according to GQM template.

Analyze the Interactive Consistency Management interface
For the purpose of understanding its effectiveness in reducing the efforts of debugging models
With respect to the users' performance, correctness, efficiency, feeling of being disrupted, and satisfaction
From the perspective of software modellers
In the context of model-editors for UML Class and State-Machine diagram modelling.

Table 6.2: Goal of the Debugging user study according to GQM template.

Recall that, to address the goals of the studies, we proposed the following 10 research questions in Chapter 5:

- *RQ1: Are users of our Focus+Context Transition Editor more effective (i.e., have higher task completeness score) when developing transitions versus users of other editors?*
- *RQ2: Are users of our Focus+Context Transition Editor more efficient (i.e., spend less time on tasks) when developing transitions versus users of other editors?*
- *RQ3: Do users of our Focus+Context Transition Editor deliver more correct solutions when writing transition expressions (i.e., develop more precise transition expressions) versus users of other editors?*
- *RQ4: Does our Focus+Context Transition Editor provide a better level of user satisfaction when developing transition expressions versus other editors?*

- *RQ5: How well does our Focus+Context Transition Editor alleviate Context-related challenges?*
- *RQ6: Are users of our Interactive Consistency Management interface more effective (i.e., create more correct models successfully) versus users of other editors?*
- *RQ7: Are users of our Interactive Consistency Management interface more efficient (i.e., spend less time on tasks) in creating precise models versus users of other editors?*
- *RQ8: Are users of our Interactive Consistency Management interface more confident in the correctness of their models (i.e., solutions) versus users of other editors?*
- *RQ9: Does our Interactive Consistency Management interface provide a better level of user satisfaction versus other editors?*
- *RQ10: Do modellers who use our Interactive Consistency Management interface judge the interface as being too intrusive and disruptive to them given the increased level of correctness that it provides?*

6.1.2 Population and Sampling

The target population of the experiment was industry-level software modellers who develop models of UML-like languages. However, recruiting such industry-level modellers is always difficult. In our experiment, we used a mixture of *convenience sampling* and *purposive sampling* approaches to recruit participants who are representative enough of the target population [7].

We recruited participants by sending an email message to ECE, CS, and SE students at the University of Waterloo. The email recipients were expected to have sufficient knowledge of UML tools as well as UML Class and State-Machine diagrams. The following two steps were taken to ensure that our subjects are as representative as possible of the target population of UML modellers: 1) We asked prospective subjects about their competency in creating UML Class and State-Machine diagrams; 2) We asked subjects to answer 10 UML-specific questions and proceeded only with those subjects who could answer all of the 10 questions. During six months of advertisement, we recruited 18 participants who fit our requirements. All the 18 subjects participated in both the Context and the Debugging studies. A summary of the subjects' occupations, familiarity with UML and experience with UML modelling tools is provided in Table 6.3. Note that we allowed the participants to

Table 6.3: Subjects' demographics and backgrounds.

Category	**Sub-Category**	**Count**
Occupation	Graduate Student	14
	Post-Doc Researcher	1
	Software Engineer (Industry)	3
UML Familiarity	Fairly Familiar (Novice)	1
	Familiar	6
	Very Familiar	8
	Strongly Familiar (Experienced)	3
Experience with Model Editors	One to six months	6
	Seven to 12 months	5
	One year to two years	2
	More than two years	5

be among the subjects who participated in our formative user study (described in Chapter 4) because of the small number of individuals who were interested in participating to the user study. Although there were overlaps on the application domain, the Class diagram, and the provided pieces of the State-Machine diagrams, we do not consider this an issue since there was a one-year gap between the formative study and this user study. Moreover, none of the questions were the same in the user studies.

6.1.3 Compensation

In appreciation of the subjects' time and commitment, they received an honorarium of $25 per session ($50 for the two user studies) and were given the chance to enter into a draw and win a prize of a $200 gift card.

6.1.4 Application Domain

We chose a fairly simple Gated Parking Lot system as the application domain in order to mitigate against the effects of domain knowledge on the participants' performance. Moreover, the participants were given the Class diagram of the system in advance and were asked to study textual descriptions of the Parking Lot domain, the classes, and their properties (i.e., attributes and operations) to become familiar with the system before the studies began.

Table 6.4: Number of participants per tool.

Our tool	Capella	VisualParadigm	MagicDraw	Papyrus
9	3	2	2	2

6.1.5 Treatment Allocation

We employed a randomized Between-Subjects [66] strategy to assign tools to participants. We selected two participants from the list of our participants (to make a pair) and then randomly selected one participant in the pair to work with a modelling tool of their own choice. The other participant in the pair was then assigned to work with our tool. We chose not to compare our tool against a single tool and instead decided to span over multiple tools because: 1) selecting a most-fitting tool, if any, as the reference tool was not straightforward, 2) it would be difficult to find participants that have enough experience of using the most-fitting tool, and 3) the resulting participant would be less reflective of the target population where the users may choose their working tool based on their experience and background knowledge.

The tool subjects' choices made in the Context study applied also to the Debugging study. Table 6.4 shows the distribution of tools used by subjects.

6.1.6 Task Design

We designed nine tasks for the Context user study and eight tasks for the Debugging user study. Each participant was given structured descriptions of the tasks and was asked to perform the tasks accordingly. The Context-related tasks included editing the Class and State-Machine diagrams of a Parking Lot system. The model-editing tasks ask the subject to set the event, guard or action segments of the transitions so that we can evaluate the level of difficulty that the subject faces with respect to recollecting the Class diagram's elements and gauge the amount of help that is provided by the tool to the subject. For example:
Task 3: Open the State Machine for Gate D. Find the transition that is labelled as Tran3. Set the Event, Guard, or Action for Tran3 based on the following description.

- *Event: After five seconds (recall that the time object notifies the system every five seconds).*
- *Guard: The gate's sensor does NOT sense blockage.*

- *Action: The gate should become closed; that is, the gate's position should be set to down.*

The Debugging-related tasks involved editing parts of the model and resolving any inconsistencies that may occur as a result of that edit. Each task was designed to introduce a different type of inconsistency (e.g., type error, undefined element error), which could help us gauge the level of aids that the tools provide to the users to resolve the errors. For example:
Task 5: The Sensor class has a Blockage attribute with the type of int. Change the Blockage attribute's type from int to bool, and fix any inconsistencies that edit may cause.

Some of the debugging tasks were designed to target type-related errors, whereas some other tasks were designed to cause use-of-undefined-elements errors. Moreover, while some of the debugging tasks were related to finding pre-existing errors in the edited diagram, other debugging tasks involved model edits that cause errors in diagrams other than the one edited.

We designed the tasks to become more complex from the first to the last task in order to support the users' learning curve, as prescribed by Pietron et al. [78].

6.1.7 Data Collection Techniques and Design

We collected several metrics to answer our research questions listed in Chapter 5, as explained below.

6.1.7.1 Performance Metrics

We measured three performance metrics as described below.

- **Task Completeness (Success score):** It is a universal metric that shows how effective a subject was in completing a task [7]. It provides compelling evidence that there is something wrong if a subject cannot successfully finish their task. We used Level of Success [7] to measure the degree to which a subject was effective in completing a task. We defined three 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

are 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 completed a task but received help during the task, such as assistance with the language syntax or clarification about a model element in the application domain.

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*: This is a traditional metric in the usability literature [7] that reports the mean average of the success scores for all of the subjects. More specifically, it refers to the summation of the subjects' score of completion 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). For instance, if 15 out of 20 subjects successfully completed the *Task 1* and 5 other subjects received help during the task (i.e., partially completed), then the *generally complete* score for the task is calculated as $(15 * 1.0) + (5 * 0.5) = 17.5$.

- **Errors:** We measured the number of errors as well. We classified the errors based on a taxonomy of Well-formedness and Consistency types of errors presented by Lange *et al.* [54]:

  – *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.

  For each subject, we counted the number of errors of each error type.

- **Efficiency:** We also measured the level of efficiency of the subjects per task. In our study, we used Lostness and Time-on-Task metrics, which are described below.

1. *Lostness* measures how *"lost"* a subject is when performing a task. The following three factors contribute to an assessment of lostness: 1) R: the minimum number of diagrams or dialogues that must be visited to accomplish a task, 2) S: the total number of diagrams or dialogue-boxes visited while performing a task (counting revisits), and 3) N: the number of different (unique) diagrams or dialogue-boxes the subject visited while performing a task. Lostness, L, is then calculated using the following formula [7][101].

   $$L = \sqrt{(\frac{N}{S} - 1)^2 + (\frac{R}{N} - 1)^2} \tag{6.1}$$

   Lostness scores range from *Zero* to *One*. The higher the score, the more trouble the user had finding what they want. Smith [101] 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 relative lostness by comparing a subject's lostness score to the optimal score (e.g., to the smallest value of lostness among all the subjects) [7].

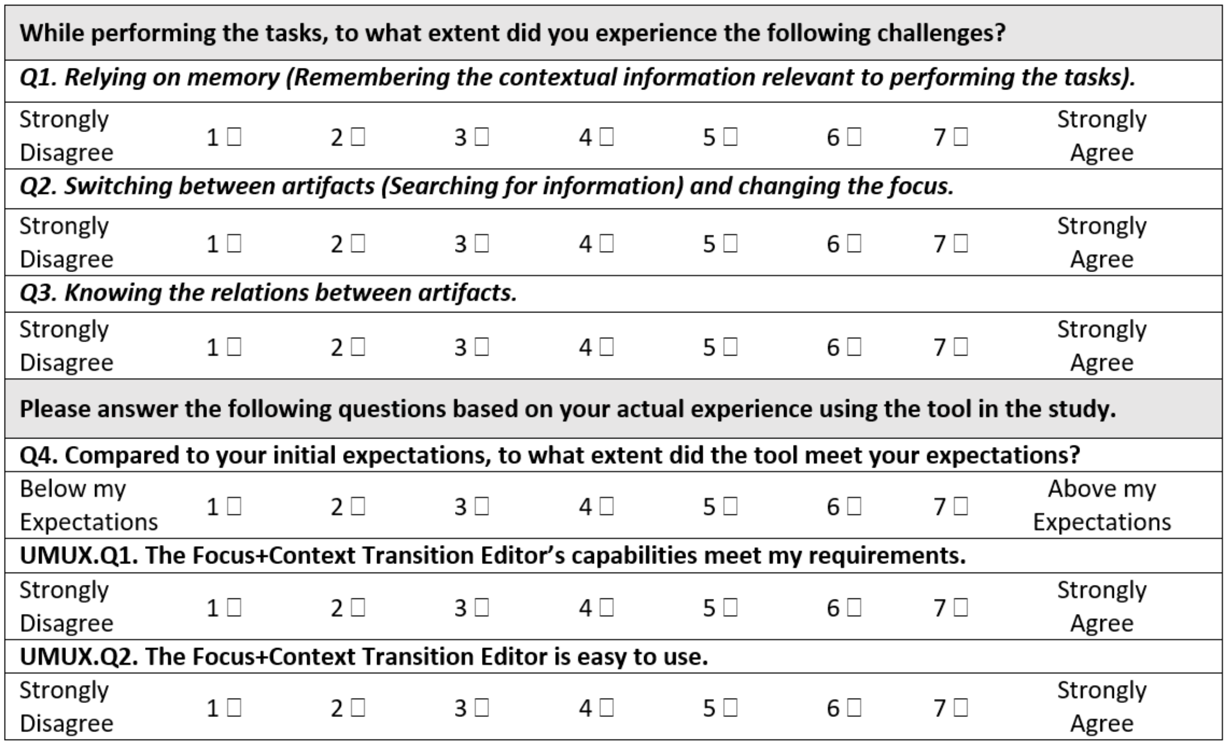
2. *Time-on-Task*: is the most popular way of measuring efficiency. It refers to the time that it takes a user to perform a particular task. In addition, it can also be thought of as the modelling effort based on Hill's definition of the modelling effort [43] described in Chapter 2.

6.1.7.2 Self-Reported Metrics

We asked the subjects about their opinion of their anticipated and perceived actual performance with respect to various aspects of using modelling tools. Specifically, we asked the subjects to rate their *satisfaction*, *experience with Context-related challenges*, *perceived confidence*, and *perceived intrusiveness*, based on the Likert scale ranging from 1 (Strongly Disagree) to 7 (Strongly Agree):

1. **Users' Satisfaction**: To measure users' satisfaction with their modelling tool, we used 1) the Usability Metric for User Experience (UMUX-Lite[8]) usability questionnaires, and 2) expectation and experience ratings.

[8]We used UMUX-Lite instead of the CSUQ (which we used in the formative study) because UMUX-Lite is a much shorter questionnaire that measures perceived usability with a non-significant discrepancy from the longer versions of usability questionnaires such as CSUQ, and is suggested by practitioners [62].



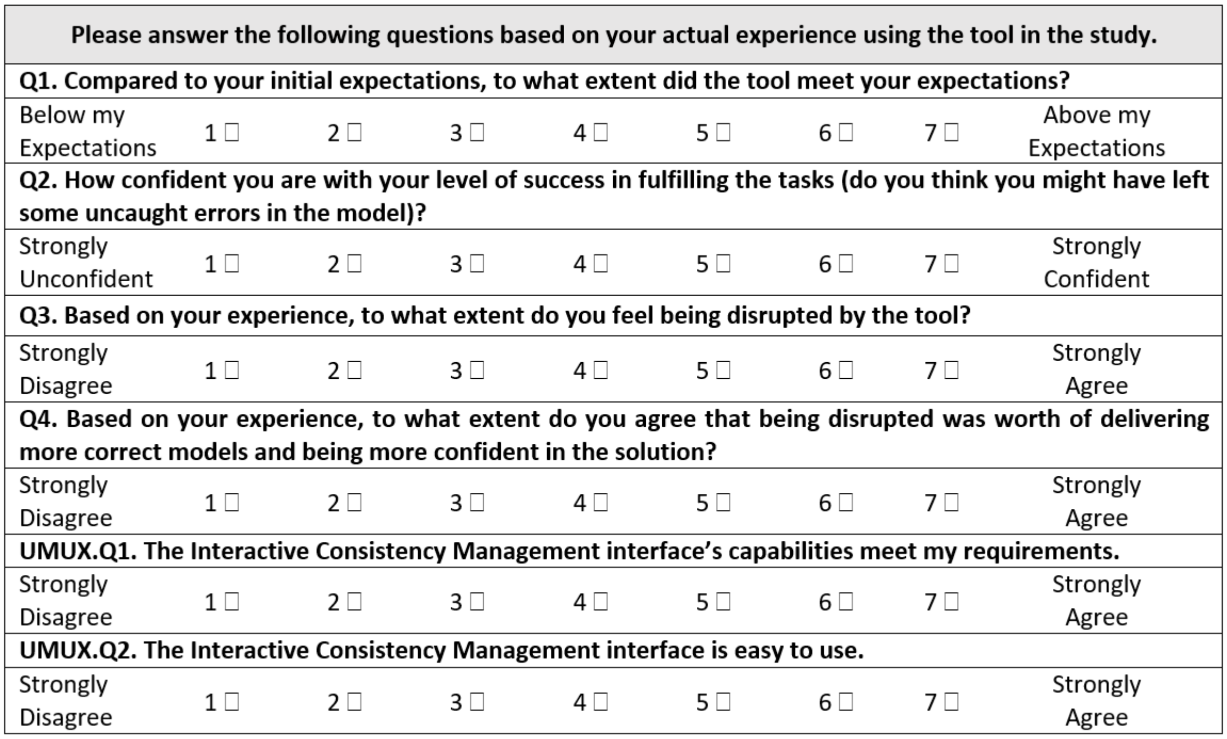
While performing the tasks, to what extent did you experience the following challenges?								
Q1. Relying on memory (Remembering the contextual information relevant to performing the tasks).								
Strongly Disagree	1 ☐	2 ☐	3 ☐	4 ☐	5 ☐	6 ☐	7 ☐	Strongly Agree
Q2. Switching between artifacts (Searching for information) and changing the focus.								
Strongly Disagree	1 ☐	2 ☐	3 ☐	4 ☐	5 ☐	6 ☐	7 ☐	Strongly Agree
Q3. Knowing the relations between artifacts.								
Strongly Disagree	1 ☐	2 ☐	3 ☐	4 ☐	5 ☐	6 ☐	7 ☐	Strongly Agree
Please answer the following questions based on your actual experience using the tool in the study.								
Q4. Compared to your initial expectations, to what extent did the tool meet your expectations?								
Below my Expectations	1 ☐	2 ☐	3 ☐	4 ☐	5 ☐	6 ☐	7 ☐	Above my Expectations
UMUX.Q1. The Focus+Context Transition Editor's capabilities meet my requirements.								
Strongly Disagree	1 ☐	2 ☐	3 ☐	4 ☐	5 ☐	6 ☐	7 ☐	Strongly Agree
UMUX.Q2. The Focus+Context Transition Editor is easy to use.								
Strongly Disagree	1 ☐	2 ☐	3 ☐	4 ☐	5 ☐	6 ☐	7 ☐	Strongly Agree

Figure 6.1: Post-session experience ratings questions for the Context study.

The UMUX-Lite includes two questions: 1) "[This system's] capabilities met my requirements." and 2) "[This system] is easy to use.". After each session, the subjects were asked to answer the UMUX questions based on a 7-point Likert scale (see questions UMUX.Q1 and UMUX.Q2 in Fig. 6.1 and Fig. 6.2).

Another traditional way of measuring the users' satisfaction is to collect pre- and post-study data about how a tool meets their expectations of using the tool. We collected and analyzed *pre-task expectation ratings* and *post-task experience ratings* and compared them against each other. Specifically, we asked the subjects to read the descriptions of all the tasks in advance and rate their *expectations* of how easy or difficult each task can be. These expectation ratings were later compared to the subjects' ratings of the level of difficulty that they actually experienced when performing the task, to gauge the gap between their expectation and experience ratings.

2. **Experienced Context-Related Challenges**: At the end of each session of the



Please answer the following questions based on your actual experience using the tool in the study.								
Q1. Compared to your initial expectations, to what extent did the tool meet your expectations?								
Below my Expectations	1 ☐	2 ☐	3 ☐	4 ☐	5 ☐	6 ☐	7 ☐	Above my Expectations
Q2. How confident you are with your level of success in fulfilling the tasks (do you think you might have left some uncaught errors in the model)?								
Strongly Unconfident	1 ☐	2 ☐	3 ☐	4 ☐	5 ☐	6 ☐	7 ☐	Strongly Confident
Q3. Based on your experience, to what extent do you feel being disrupted by the tool?								
Strongly Disagree	1 ☐	2 ☐	3 ☐	4 ☐	5 ☐	6 ☐	7 ☐	Strongly Agree
Q4. Based on your experience, to what extent do you agree that being disrupted was worth of delivering more correct models and being more confident in the solution?								
Strongly Disagree	1 ☐	2 ☐	3 ☐	4 ☐	5 ☐	6 ☐	7 ☐	Strongly Agree
UMUX.Q1. The Interactive Consistency Management interface's capabilities meet my requirements.								
Strongly Disagree	1 ☐	2 ☐	3 ☐	4 ☐	5 ☐	6 ☐	7 ☐	Strongly Agree
UMUX.Q2. The Interactive Consistency Management interface is easy to use.								
Strongly Disagree	1 ☐	2 ☐	3 ☐	4 ☐	5 ☐	6 ☐	7 ☐	Strongly Agree

Figure 6.2: Post-session experience ratings questions for the Debugging study.

Context study, we asked the subjects to what extent they experienced Context-related challenges when performing tasks (see Q1, Q2, and Q3 in Fig. 6.1).

3. **Perceived Confidence**: For the Debugging study, we asked users' to rate how confident they were with their level of success in fulfilling the debugging tasks (i.e., to what extent they thought they had left any uncaught errors in the model).

4. **Perceived Intrusiveness**: To measure how intrusive our Consistency Management interface is, we asked the subjects who used our tool, before the Debugging sessions. The *pre-session expectation* questions asked the subjects to rank on a Likert scale (from 1=Strongly Disagree to 7=Strongly Agree) the extent to which they agreed that maintaining model consistency on-the-fly while model editing is counter-productive (disruptive when performing modelling tasks). The questions were:

   - 1) To what extent do you agree that inconsistencies should be prevented immediately rather than allowing them to exist and fix them at a later time?

- 2) To what extent do you agree that maintaining inconsistencies at all time is counter-productive and intrusive (disrupting the focus tasks)?

At the end of the Debugging study, we asked subjects to rank the extent to which they felt that the interruptions by the Consistency Management editor were disruptive (see Q3 and Q4 in Fig. 6.2).

6.2 Execution and Practical Considerations

We conducted the user studies in 36 separate sessions: 18 sessions per study (two sessions per subject) and nearly 90 minutes per session. A subject could decide if they preferred to participate in either of the Context study or the Debugging study, or both. All subjects were eager to participate in both of the user studies. The order of the studies was not deemed to be significant; however, we decided to conduct the Context user study before the Debugging study because we wanted all the subjects to take the same procedure instead of letting them decide about the order of the studies. Each session was scheduled separately based on the availability of the researcher and the subject. A subject could choose if they wanted to have their Debugging session right after their Context session, or if they preferred to postpone their Debugging session to a later time but within two weeks of their Context session. In our case, all the subjects preferred to perform the Debugging session right after the Context session. However, they could have a 15-minutes break between the sessions to refresh themselves.

The subjects were asked to perform their tasks using a Windows desktop in the researcher's office. We designed and used macro-based Microsoft Excel workbooks to automate the process of data-collection. Furthermore, we designed an individual sheet for each task, which included the task's description and the post-task rating questions related to that task. For example, there were nine sheets corresponding to the nine tasks of the Context study. For each task, the subject was supposed to open the corresponding sheet of the task, read the task's description, perform the task using the tool, and answer the post-task ratings of the task. Once they were done with a task, they could move on to the next sheet and perform the same procedure for the next task. The time on each task was automatically recorded by a macro that started a timer when the subject opened the task's sheet and ended the timer when the subject switched to the next sheet (i.e., next task). By doing so, we could collect and store all types of data, such as the subject's times on tasks and responses to the expectation and experience questionnaires, using the excel files.

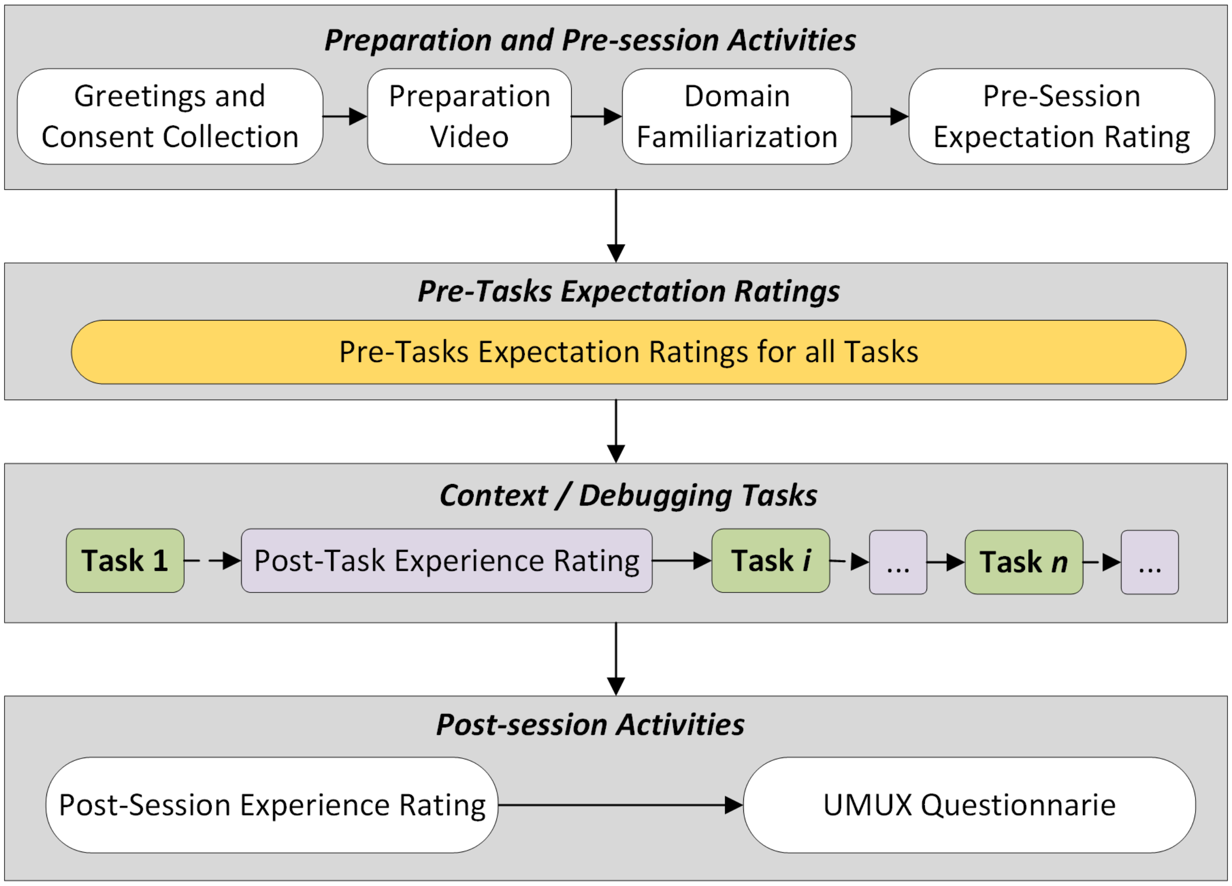

Preparation and Pre-session Activities
Greetings and Consent Collection
Preparation Video
Domain Familiarization
Pre-Session Expectation Rating
Pre-Tasks Expectation Ratings
Pre-Tasks Expectation Ratings for all Tasks
Context / Debugging Tasks
Task 1
Post-Task Experience Rating
Task i
...
Task n
...
Post-session Activities
Post-Session Experience Rating
UMUX Questionnarie

Figure 6.3: The work-flow of a subject's session.

As shown in Fig. 6.3, the studies consisted of four main segments: Preparation and Pre-session Activities, Collecting Pre-Task Expectation Ratings, Performing the Tasks, and Post-Session Activities. We explain these segments as follows.

6.2.1 Preparation and Pre-session Activities

Each session started with a greeting to convey an informal atmosphere for the subject. We wanted the subjects to be relaxed and comfortable, so that they would perform the study tasks as effective as possible. We asked them to read and sign the consent form, and ensured that they knew their rights during the session.

As a second step, each subject was asked to view a preparation video which aimed to familiarize them with the goal and methods of the study (e.g., think-out-loud protocol). The video made it clear that it was the tools that were under scrutiny and not them. The main rationale for using a video rather than in-person preparation was to mitigate the risks of providing different information and introductions to different subjects.

To gain knowledge about the application domain, the subjects were given a textual description of the Gated Parking-Lot domain and were asked to read it carefully. Moreover, the domain description included the Class diagram and descriptions of all of the diagram's model elements (e.g., classes, operations, attributes). Since we used the same application domain for both of the Context and Debugging studies, we offered the description only before the Context study to save some time during the Debugging sessions. However, the subjects were able to refer to the description at any point during the study.

Before each Debugging session, we asked the subjects to answer (based on a Likert scale) a few pre-session expectation questions about their views on whether it would be disruptive to be warned about and required to resolve model errors as they occur during model editing. These questions were:

- *To what extent do you agree that inconsistencies should be resolved immediately rather than allowing them to exist and fixing them at a later time?*, and
- *To what extent do you agree that maintaining model consistency at all times is counter-productive and intrusive (disrupting the focus tasks)?*

6.2.2 Collecting Pre-Task Expectations Ratings and Performing the Tasks

Each task was preceded by a pre-task expectation rating. A pre-task expectation rating asked the subject to read a task's description and estimate how easy or difficult they thought the task would be. We did all the pre-task ratings at the beginning of the session because we wanted to avoid the learning effect that performing a task could have on the ratings of the next task.

After collecting the pre-session ratings, all the subjects went through a "warm-up" phase before each Context and Debugging session, in which the researcher and the subject walked through the tool and practiced a mock exercise (in the Parking-Lot domain) in order to help the subject recall the relevant tools' features and become ready to use the tool. During the session, the subject was required to use their assigned or chosen tool (depending on whether they were assigned to use our tool or were assigned to use the existing UML model editor of their choice) to perform the tasks. Moreover, the subject was asked to express their thoughts out-loud during the session, to enable us capture and understand their cognitive challenges. We recorded the subjects' voice and screen-captured their work with the tool for later analysis.

Furthermore, each task was followed by a post-task experience rating. Post-task experience rating asked the subject to rank the level of difficulty that they actually experienced when performing the task.

6.2.3 Post-session Activities

At the end of each session, the subject was asked to answer the post-session experience ratings and UMUX satisfaction questionnaires. While the UMUX questions were the same between the Context and Debugging studies, the Post-Session Experience Ratings questions varied between the two studies because the goals of the studies were different.

6.2.4 Practical Considerations

Motivating subjects to fulfil a task in a timely manner while performing at their best and providing high-quality solutions is a delicate balancing act. If we were to impose time constraints on the subjects, it can adversely impact the quality of their solutions. Thus, we allowed the subjects to perform at their own pace and self-announce the completion of a task. The downside of this approach is that a subject could spend an excessive amount of time performing a task leading to unrealistically long and useless measurements of times on the tasks. As an attempt to achieve balance between good performance on tasks and completion of all tasks, we applied the following considerations:

- We allocated a no-progress limit of ten minutes for each task. That is, if a subject did not show any sign of progress within ten minutes, they would be interrupted by the researcher and asked if they needed any assistance (e.g., *"Do you need any help or do you have any question that I can help with?"*). If the subject asked for the help, then the researcher would provide a hint to fulfil the task and the subject's task success would subsequently be scored between Incomplete (0) or Partially Complete (0.5). Such subjects were given another five minutes after the given hint to complete the task. If a subject was still unable to complete the task after the five minutes, then the researcher asked her to switch to the next task and the task success was considered as Incomplete (0).

- Instead of offering an hourly rate, we offered a fixed honorarium of $25 in return for each session. Hence, the subject knew that they would receive the same amount of compensation even if they completed their tasks earlier than expected. This also avoided the risk of subjects playing around with the tool to receive a higher payment.
- The subject was allowed to leave the experiment as soon as they completed the tasks.

6.3 Results

This section presents the results of the Context and Debugging user studies with respect to the aforementioned research questions. For each study, we apply the Mann Whitney U test to analyse the effects of using our tool versus using other tools. We use Mann-Whitney U test for the quantitative analysis of our results over other tests such as two-sample t-test because of its strength in producing more precise results in cases of small numbers of participants and its nature of not requiring normally distributed samples [100]. All tests were performed using Excel's statistical package under the criteria of having a significance level of 0.05 (i.e., 95% confidence interval). Based on the results of the statistical tests, we can affirm that all effects were statistically significant at the $\alpha = 0.05$ significance level.

6.3.1 Results of the Context User Study

The Context user study was designed to assess our Focus+Context Transition Editor against five research questions, as described below.

6.3.1.1 RQ1: Are users of our Focus+Context Transition Editor more effective (i.e., have higher task completeness score) when developing transitions versus users of other editors?

Task Completeness (Success): For each task, we computed a *mean* score for all subjects on that task. The mean success score for all the participants, shown in Fig. 6.4, demonstrates that users of our tool scored higher success scores versus users of the other editors for all the tasks. It also shows that the difference in the success scores of the two groups of users is much higher when the task's difficulty is higher (see tasks 8 and 9). The users' cognitive effort is higher when the task difficulty is higher; in these cases, the extra tooling support might play a more important role in improving users' performances.

Table 6.5: Context Study: Results of each subject's success in completing each task

Participant	Task1	Task2	Task3	Task4	Task5	Task6	Task7	Task8	Task9
P1	0.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0
P2	1.0	1.0	1.0	0.5	1.0	1.0	1.0	1.0	0.0
P3	0.5	1.0	0.5	0.5	1.0	0.0	0.5	0.0	0.0
P4	1.0	1.0	0.5	1.0	1.0	1.0	1.0	0.5	0.0
P5 *	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5
P6	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P7 *	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P8 *	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5
P9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	1.0
P10 *	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P11 *	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P12 *	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P13 *	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P14 *	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P15 *	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P16	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0
P17	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5
P18	1.0	1.0	1.0	1.0	1.0	0.5	1.0	1.0	1.0
mean	0.89	1.00	0.89	0.89	1.00	0.83	0.94	0.78	0.39
median	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
stdev	0.22	0.00	0.22	0.22	0.00	0.35	0.17	0.36	0.49
mean*	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.89
median*	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
stdev*	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.22

* Denotes the subjects who used our tool.

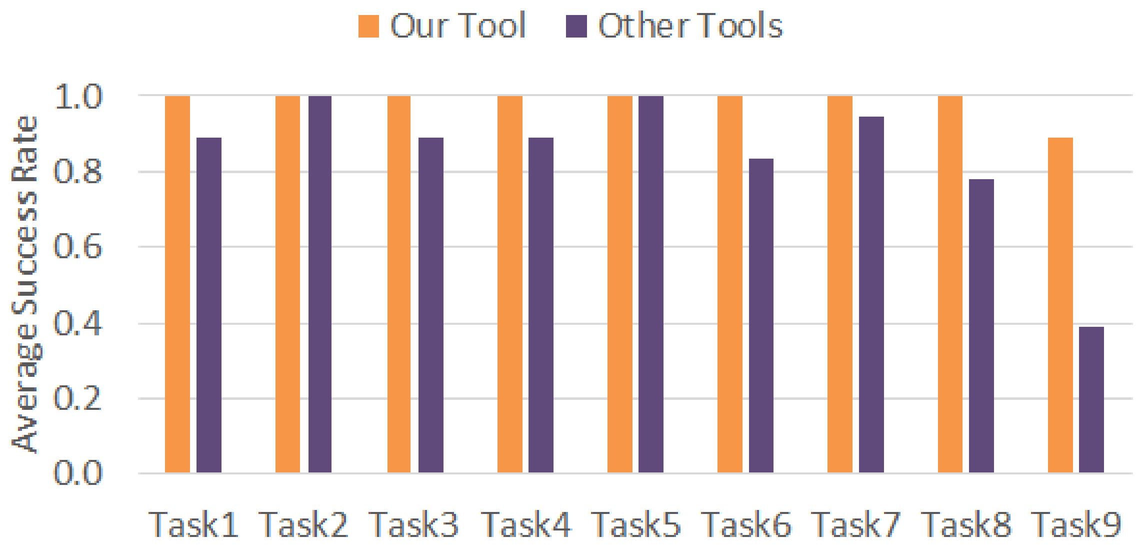


Figure 6.4: Context Study: Mean success rate per task for users of our editor vs. users of other editors.

Another interesting observation from the results is that for Task 6, where the subjects only needed to copy/reuse elements of an existing transition in another transition, there are two subjects whose success score is not 1 (i.e., complete success). We found that the user may forget what they just memorized (i.e., the original transition's elements) and perform their task based on their incomplete memory. While this incomplete memory recall can lead to some small (but important) errors such as misspelling an element's name, it can also lead to major errors such as writing wrong navigation expressions (e.g., writing *gate.transponder* instead of *gate.sensor.transponder*). The results of the subjects' success scores in completing the tasks are presented in Table 6.5. Please note that the order of participants shown in the table (and the following tables) does not reflect their order of participation in the study. The order shown here is based on the order of analysis we have done on the subjects.

6.3.1.2 RQ2: Are users of our Focus+Context Transition Editor more efficient (i.e., spend less time on tasks) when developing transitions versus users of other editors?

Time on Task: Fig. 6.5 shows the mean time on each task for users of our editor versus the mean time on task for the users of other editors. For a more meaningful result, we

Table 6.6: Context Study: Results of the subjects' time on each task (in minutes).

Participant	Task1	Task2	Task3	Task4	Task5	Task6	Task7	Task8	Task9
P1	04:42	04:11	04:10	05:10	02:19	02:13	03:06	01:54	16:06
P2	08:43	07:39	03:14	02:07	00:44	02:16	02:57	02:36	18:40
P3	04:12	03:25	05:11	02:23	01:37	05:13	06:48	04:31	11:00
P4	06:40	02:59	06:21	01:57	00:23	04:07	02:50	08:41	12:48
P5 *	04:08	06:06	02:03	01:51	00:56	00:50	02:34	02:32	26:24
P6	02:59	02:47	03:27	01:43	01:44	02:19	02:46	02:22	15:09
P7 *	02:27	04:49	02:36	03:30	01:17	01:30	02:00	04:46	11:06
P8 *	02:41	02:40	03:02	02:21	00:57	01:47	02:38	02:05	11:45
P9	10:12	03:50	03:36	05:45	02:07	04:09	01:43	03:55	12:34
P10 *	02:54	01:55	05:08	02:52	00:50	01:58	03:19	02:34	13:30
P11 *	03:44	07:49	05:40	05:25	01:02	02:49	04:49	04:31	19:18
P12 *	02:14	04:55	02:24	02:09	00:47	01:27	02:29	02:32	09:41
P13 *	02:55	03:59	02:33	01:44	01:10	01:12	02:31	02:14	10:51
P14 *	01:55	03:08	01:25	01:38	00:45	00:49	02:30	01:57	09:46
P15 *	02:10	04:02	01:58	03:11	02:29	01:40	02:43	02:03	13:19
P16	09:37	07:06	06:21	04:21	01:36	05:36	04:48	04:19	14:20
P17	06:18	04:16	04:24	02:32	01:46	02:34	04:52	03:49	13:21
P18	08:04	04:23	04:46	02:34	01:53	03:11	04:50	02:56	17:36
geo-mean	06:22	04:16	04:29	02:54	01:24	03:18	03:34	03:32	14:26
median	06:40	04:06	03:32	02:27	01:13	02:15	02:48	02:35	13:20
stdev	02:31	01:43	01:32	01:20	00:36	01:25	01:19	01:40	04:10
geo-mean*	02:43	04:03	02:43	02:33	01:04	01:27	02:45	02:39	13:13
median*	02:41	04:02	02:33	02:21	00:57	01:30	02:34	02:32	11:45
stdev*	00:44	01:48	01:27	01:12	00:32	00:37	00:49	01:04	05:30

* Denotes the subjects who used our tool.

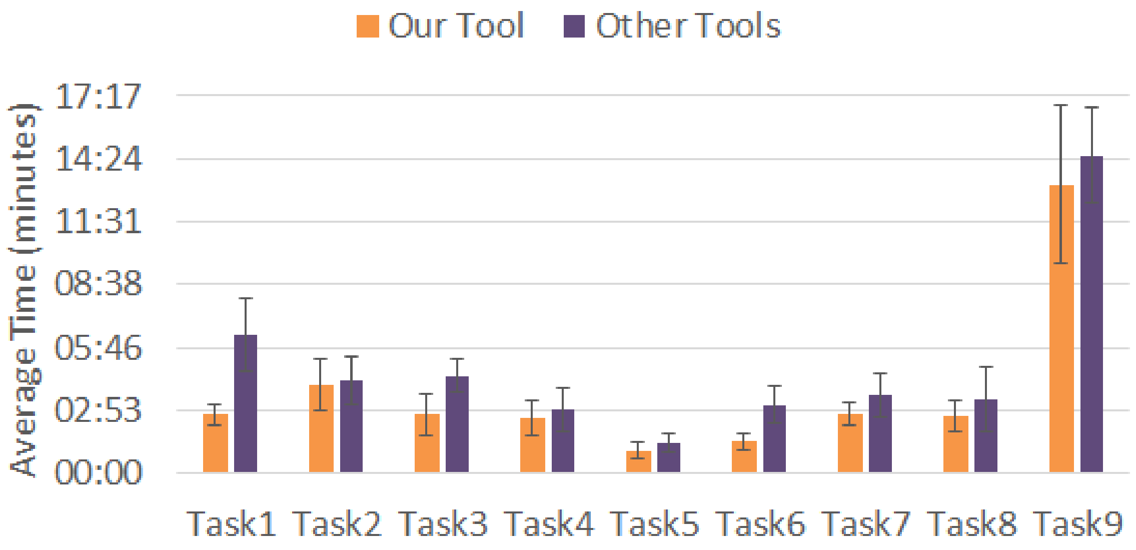


Figure 6.5: Context Study: Average time per task for users of our editor vs. users of other editors.

included in the average only the times on completely successful and partially successful tasks. As shown in the figure, the subjects who used our tool spent less time to perform the tasks than the users of the other editors.

Table 6.6 presents the results of the times on tasks for all the subjects for the Context study. As shown in the table, the time spent on Task 6 for the users who used our editor is much less than the time spent on the task for the users of other editors. The only reason for this average time difference is that users of other tools used the copy and paste feature of the tools but the users of our tool took advantage of the reuse feature that we embedded in our transition editor (recall that this task asked the users to reuse elements of an existing transition). For the other tasks, we believe that the improvement in the times on the tasks is because of the sliced view of the contextual information, the search capability, and the drag-n-drop mechanism that are provided by our editor. Furthermore, it can be noticed that among the users who used other tools, P6 performed better than others. We found that P6 declared himself to be a Senior Software Engineer in the industry with a high level of competency with UML and more than two years of experience with UML tools.

Table 6.7: Context Study: Lostness results for the tasks for 18 subjects.

Participant	Task1		Task2		Task3		Task4		Task5		Task6		Task7		Task8		Task9	
	R,N,S	L	R,N,S	L	R,N,S	L	R,N,S	L	R,N,S	L	R,N,S	L	R,N,S	L	R,N,S	L	R,N,S	L
P1	1,2,4	0.71	1,2,5	0.78	1,2,3	0.60	1,2,9	0.92	1,1,1	0.00	1,2,7	0.87	1,2,2	0.50	2,3,4	0.42	2,2,4	0.50
P2	1,2,11	0.96	1,3,3	0.67	1,2,3	0.60	1,2,3	0.60	1,2,3	0.60	1,3,6	0.83	1,2,3	0.60	2,3,3	0.33	2,6,33	1.00
P3	1,1,1	0.00	1,1,1	0.00	1,2,4	0.71	1,1,1	0.00	1,1,1	0.00	1,2,5	0.78	1,1,1	0.00	2,1,1	0.00	2,2,5	0.60
P4	1,2,2	0.50	1,3,3	0.67	1,2,2	0.50	1,2,7	0.87	1,3,3	0.67	1,3,10	0.97	1,2,5	0.78	2,2,3	0.33	2,4,12	0.83
P5 *	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	2,1,1	0.00	2,2,4	0.50
P6	1,2,3	0.60	1,2,5	0.78	1,2,3	0.60	1,2,3	0.60	1,2,3	0.60	1,2,3	0.60	1,2,3	0.60	2,3,4	0.42	2,4,8	0.71
P7 *	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	2,2,2	0.00	2,2,2	0.00
P8 *	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	2,1,1	0.00	2,2,2	0.00
P9	1,2,5	0.78	1,1,1	0.00	1,2,5	0.78	1,2,5	0.78	1,1,1	0.00	1,2,11	0.96	1,2,3	0.60	2,3,4	0.42	2,3,6	0.60
P10 *	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	2,2,2	0.00	2,2,2	0.00
P11 *	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	2,2,2	0.00	2,2,2	0.00
P12 *	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	2,2,2	0.00	2,2,2	0.00
P13 *	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	2,2,2	0.00	2,2,4	0.50
P14 *	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	2,2,2	0.00	2,2,2	0.00
P15 *	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	2,2,2	0.00	2,2,2	0.00
P16	1,2,5	0.78	1,2,7	0.87	1,2,9	0.92	1,2,3	0.60	1,2,3	0.60	1,2,7	0.87	1,2,5	0.78	2,3,3	0.33	2,1,1	0.00
P17	1,2,7	0.87	1,2,5	0.78	1,2,5	0.78	1,2,5	0.78	1,2,3	0.60	1,2,3	0.60	1,2,3	0.60	2,3,4	0.42	2,3,8	0.71
P18	1,2,5	0.78	1,2,5	0.78	1,1,1	0.00	1,2,5	0.78	1,1,1	0.00	1,2,2	0.50	1,2,3	0.60	2,3,4	0.42	2,3,12	0.82
mean		0.66		0.59		0.61		0.66		0.34		0.78		0.56		0.34		0.64
median		0.78		0.78		0.60		0.78		0.60		0.83		0.60		0.42		0.71
stdev		0.28		0.34		0.26		0.27		0.32		0.17		0.23		0.13		0.28
efficient subj. (< 0.5)		11.11%		22.22%		11.11%		11.11%		44.44%		0.00%		11.11%		100.00%		11.11%
inefficient subj. (>= 0.5)		88.89%		77.78%		88.89%		88.89%		55.56%		100.00%		88.89%		0.00%		88.89%
mean *		0.00		0.00		0.00		0.00		0.00		0.00		0.00		0.00		0.11
median*		0.00		0.00		0.00		0.00		0.00		0.00		0.00		0.00		0.00
stdev*		0.00		0.00		0.00		0.00		0.00		0.00		0.00		0.00		0.22
efficient subj. (< 0.5) *		100.00%		100.00%		100.00%		100.00%		100.00%		100.00%		100.00%		100.00%		77.78%
inefficient subj. (>= 0.5) *		0.00%		0.00%		0.00%		0.00%		0.00%		0.00%		0.00%		0.00%		22.22%

* Denotes the subjects who used our tool.

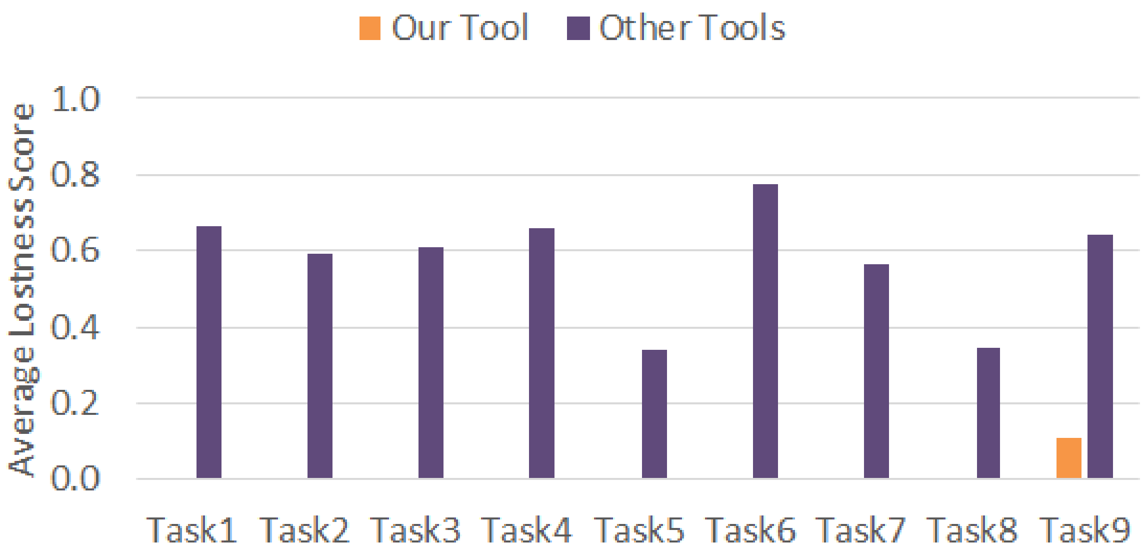


Figure 6.6: Context Study: Average lostness score per task for users of our editor vs. users of other editors.

Lostness: Fig. 6.6 shows the average of the subjects' lostness scores broken down by task. The participants who used other editors were *"lost"* in all of the tasks (except for tasks 5 and 8), whereas the users of our editor showed almost no sign of being lost (except for Task 9). This is mostly because of the contextual view that is provided by our tool, which reduced the users' need to switch back and forth among different diagrams to recollect contextual information relevant to the tasks. For Task 9, which was a more difficult task, users needed a more comprehensive view of the Class diagram rather than just a minimal sliced view to make more reasonable and more thorough changes. Thus, the lostness score (slightly) increased for the users of our editor compared to the previous eight tasks. However, it is still much less than the average lostness score obtained from the users of other editors. The result of the lostness scores is presented in Table 6.7. The table shows the percentage of *lost* subjects (i.e., inefficient subjects) who used other tools versus the percentage of *lost* subjects who used our tool. Moreover, one can notice that for a more complex task (i.e., Task 9), the users of our tool also showed signs of lostness. This suggests that our tool may need more thorough enhancements to cope with the task complexity.

Table 6.8: Context Study: Result of the subjects' errors on each task.

Participant	Task1	Task2	Task3	Task4	Task5	Task6	Task7	Task8	Task9
P1	1	1	0	1	0	0	0	0	0
P2	3	1	2	1	1	0	0	0	3
P3	1	3	2	2	1	2	0	0	0
P4	2	1	1	2	1	0	1	0	0
P5 *	0	0	0	0	0	0	0	0	0
P6	2	2	1	1	0	0	0	0	4
P7 *	0	0	0	0	0	0	0	0	0
P8 *	0	0	0	0	0	0	0	0	0
P9	2	1	1	0	1	0	0	0	1
P10 *	0	0	0	0	0	0	0	0	0
P11 *	0	0	0	0	0	0	0	0	0
P12 *	0	0	0	0	0	0	0	0	0
P13 *	0	0	0	0	0	0	0	0	0
P14 *	0	0	0	0	0	0	0	0	0
P15 *	0	0	0	0	0	0	0	0	0
P16	1	0	1	0	0	0	0	0	0
P17	3	2	0	0	0	0	1	0	0
P18	2	2	1	0	2	0	0	0	2
mean	1.89	1.44	1.00	0.78	0.67	0.22	0.22	0.00	1.11
median	2.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00
stdev	0.78	0.88	0.71	0.83	0.71	0.67	0.44	0.00	1.54
mean*	0	0	0	0	0	0	0	0	0
median*	0	0	0	0	0	0	0	0	0
stdev*	0	0	0	0	0	0	0	0	0

* Denotes the subjects who used our tool.

6.3.1.3 RQ3: Do users of our Focus+Context Transition Editor deliver more correct solutions when writing transition expressions (i.e., develop more precise transition expressions) versus users of other editors?

Task Errors: The results of our analysis on the subjects' errors on the tasks (see Fig. 6.7 and Table 6.8) show that the subjects who used our tool made fewer errors (in fact no errors at all) than the subjects who used other tools. The main reason to this is that our tool employs a correct-by-construction approach (and a formal language). Our tool also tries to automate most of the users' tasks which reduce the chance of slips and lapse. For instance, the users of our tool will never make a spelling error when writing the name of an element in a guard condition, whereas the users of other editor might make such mistakes because of writing their transition guards in plain text. We believe that, the result of the errors on the tasks are meant to be incorporated with the results from other research questions to become more meaningful. For instance, we will later show that the correct-by-construction can improve the correctness and at the same time behave in a way that it does not disrupt users from their tasks.

The results also show that the average number of errors made by subjects per task decreased from Task 1 to Task 8. The main reason for this is that in the later tasks we provided the subjects with more guidelines on how to create a correct model. For instance, Task 8 mentions: *"Use a function named Transponder. You may need to create this function in the Sensor class before setting it in the Event."*. Therefore, the subjects were made aware of the steps that they should take towards creating a correct transition expression.

According to the results, the subjects who used Capella (i.e., P1, P2, and P16) delivered more correct models (average of 0.59 errors for all of the tasks) versus the subjects who used other tools. The reason to this is the Content-Assist feature that is embedded in Capella, which helps in writing transition's event, guard and actions. Although their Content-Assist is a very basic one, but it showed to be effective in improving the correctness of subjects' solutions. More specifically, using the Content-Assist feature, Capella users made fewer basic errors such as spelling errors, wrong navigation expressions, or referencing to an undefined element. Similarly, the subjects who used Visual Paradigm (i.e., P9 and P17) delivered more correct solutions than the subjects who used MagicDraw and Papyrus. The reason is that, Visual Paradigm also allows the user to refer to a model element when developing transition's events and actions. Although such a feature also exists for Papyrus, it is considered to be complex using the tool (i.e., lack of usability). Therefore, users of tools such as Papyrus prefer to write the transitions in plain text, which may introduce more errors.

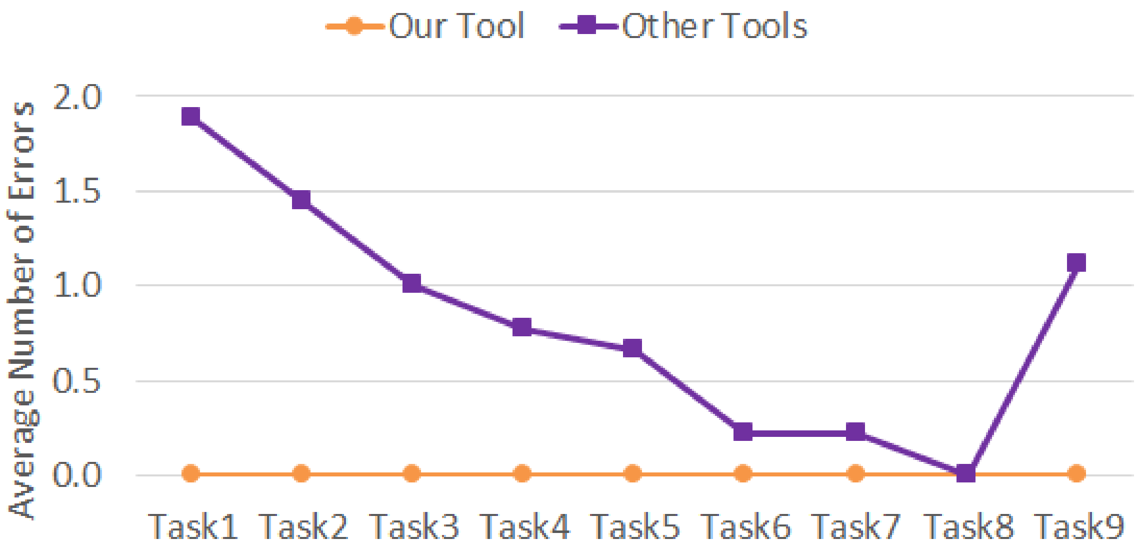


Figure 6.7: Context Study: Average number of errors per task made by users of our editor vs. users of other editors.

Table 6.9: Context Study: Post-session experience ratings.

Participant	Context (Q1)	Switch (Q2)	Relations (Q3)	Satisfaction (Q4)	Capability (UMUX.Q1)	Usability (UMUX.Q2)
P1	4	6	4	6	6	7
P2	4	6	6	6	5	5
P3	7	4	7	5	3	2
P4	5	2	0	2	4	2
P5 *	1	2	1	6	6	7
P6	6	7	6	2	2	2
P7 *	1	2	1	6	7	7
P8 *	2	1	3	5	5	5
P9	6	7	5	2	3	2
P10 *	1	2	1	6	6	6
P11 *	2	1	2	6	5	6
P12 *	2	3	2	6	6	6
P13 *	3	2	3	6	6	5
P14 *	1	1	1	7	6	6
P15 *	2	2	2	5	6	6
P16	3	4	4	3	3	3
P17	5	6	5	3	2	4
P18	6	7	7	2	2	2
mean	5.11	5.44	4.89	3.44	3.33	3.22
median	5.00	6.00	5.00	3.00	3.00	2.00
stdev	1.27	1.74	2.15	1.74	1.41	1.79
mean*	1.67	1.78	1.78	5.89	5.89	6.00
median*	2.00	2.00	2.00	6.00	6.00	6.00
stdev*	0.71	0.67	0.83	0.60	0.60	0.71

* Denotes the subjects who used our tool.

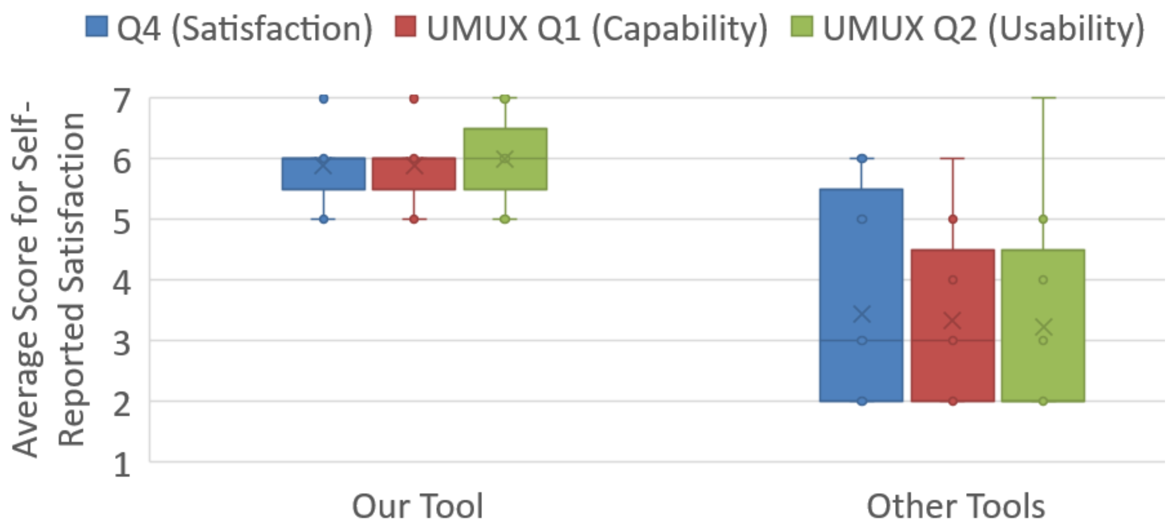


Figure 6.8: Context Study: Average satisfaction scores for users of our editor vs. users of other editors.

6.3.1.4 RQ4: Does our Focus+Context Transition Editor provide a better level of user satisfaction when developing transition expressions versus other editors?

We measured the subjects' opinions of their satisfaction by asking three post-session questions, i.e., Q4, UMUX.Q1, and UMUX.Q2 shown in Fig. 6.1. Recall that Q4 asked the users to rate whether their experience of using the tool was beyond their initial expectation or below. The UMUX.Q1 and UMUX.Q2 also asked the users to rate how well the tool capabilities met their requirements and how easy-to-use the tool was, respectively. As shown in Fig. 6.8, while the subjects who used other tools expressed a neutral level of satisfaction (i.e., mean value of 4) for the three questions, subjects who used our tool declared to be satisfied with our tool (mean value of 6). See Table 6.1 for the results.

Also, we compared the pre-task expectation ratings (see Table 6.10) and post-task experience ratings (see Table 6.11), comparing the ratings of subjects who used our tool versus the ratings of subjects who used other tools. The difference between the ratings of the two classes of subjects also show that our tool could better meet the users' expectations. One can notice that the users of other tools have lower expectation ratings of their tools than the users of our tool. Based on our observations, the subjects who used our tool rated more conservatively because they were not familiar with our tool. That is, they took into

Table 6.10: Context Study: Pre-task expectation ratings.

Participant	Task1	Task2	Task3	Task4	Task5	Task6	Task7	Task8	Task9
P1	2	2	2	2	2	2	2	2	5
P2	3	4	3	3	2	2	2	2	6
P3	1	1	1	1	1	1	1	4	6
P4	1	2	2	1	1	1	4	2	5
P5 *	2	3	1	2	2	2	4	3	7
P6	3	3	3	3	2	2	3	3	4
P7 *	3	3	4	3	3	4	5	5	6
P8 *	3	3	3	3	3	4	4	4	5
P9	2	3	3	3	2	1	6	4	6
P10 *	2	3	4	4	2	3	5	6	7
P11 *	2	3	4	2	2	4	5	4	6
P12 *	3	5	5	2	2	2	5	3	6
P13 *	3	3	4	4	2	3	5	4	6
P14 *	3	4	3	3	2	2	4	3	6
P15 *	3	4	3	2	2	5	5	3	6
P16	2	2	3	3	2	2	3	4	6
P17	2	2	2	2	1	1	3	3	6
P18	1	1	2	2	1	1	2	2	5
mean	1.89	2.22	2.33	2.22	1.56	1.44	2.89	2.89	5.44
median	2.00	2.00	2.00	2.00	2.00	1.00	3.00	3.00	6.00
stdev	0.78	0.97	0.71	0.83	0.53	0.53	1.45	0.93	0.73
mean*	2.67	3.44	3.44	2.78	2.22	3.22	4.67	3.89	6.11
median*	3.00	3.00	4.00	3.00	2.00	3.00	5.00	4.00	6.00
stdev*	0.50	0.73	1.13	0.83	0.44	1.09	0.50	1.05	0.60

* Denotes the subjects who used our tool.

Table 6.11: Context Study: Post-task experience ratings.

Participant	Task1	Task2	Task3	Task4	Task5	Task6	Task7	Task8	Task9
P1	4	4	4	6	5	2	3	3	5
P2	4	4	4	5	4	4	4	4	7
P3	2	2	4	2	1	3	3	5	7
P4	1	2	2	2	1	2	3	5	3
P5 *	1	1	1	1	1	1	1	1	5
P6	2	3	4	3	2	5	4	5	7
P7 *	1	2	1	2	1	1	2	2	4
P8 *	1	2	3	2	1	2	3	1	4
P9	4	6	4	6	5	4	7	6	7
P10 *	1	2	4	3	1	1	3	2	6
P11 *	2	3	3	2	2	2	3	3	5
P12 *	1	3	2	1	1	1	2	2	4
P13 *	1	2	1	1	1	1	2	1	3
P14 *	1	2	1	1	1	1	1	1	3
P15 *	2	3	1	2	2	1	2	1	3
P16	3	3	4	4	3	4	5	5	7
P17	3	4	3	2	1	2	5	5	7
P18	3	3	4	3	2	3	4	3	7
mean	2.89	3.44	3.67	3.67	2.67	3.22	4.22	4.56	6.33
median	3.00	3.00	4.00	3.00	2.00	3.00	4.00	5.00	7.00
stdev	1.05	1.24	0.71	1.66	1.66	1.09	1.30	1.01	1.41
mean*	1.22	2.22	1.89	1.67	1.22	1.22	2.11	1.56	4.11
median*	1.00	2.00	1.00	2.00	1.00	1.00	2.00	1.00	4.00
stdev*	0.44	0.67	1.17	0.71	0.44	0.44	0.78	0.73	1.05

* Denotes the subjects who used our tool.

account the learning curve of the tool as well. However, this unfamiliarity was not true because we performed a warm-up task afterwards. Also, we realized that a rating scale of 1 to 7 was too wide for our purpose. We can only draw a qualitative conclusion indicating that whether the user's experience were above their expectation or below. Thus, we can conclude that, in a qualitative manner, the subjects who used our tool were more satisfied based on their initial expectation of using the tool versus the subjects who used other tools. However, we cannot conclude that the users of our tool were significantly more satisfied versus the users of other tools.

Furthermore, when analysing the verbal expressions of the users, we found that having distinguished input text-boxes for each of the event, guard, and action segments of a transition could improve users' experience of developing transitions. Additionally, many of the users who used our editor expressed their satisfaction by saying *"Nice"* at the event of the task. This could be because either they were satisfied with the editor or just because of experiencing a new way of writing transitions (we do not know which one). What is important to mention is that, the number of users who made negative expressions when using our editor is less than the number of users who used other editors. This can also be due to the fact that they were excited to experience a new tool or they just preferred to be less harsh and provide more constructive feedback about the tool (because of the presence of the researcher). In contrast, the users who used other editors showed signs of frustration (e.g., sighs) at different points in their sessions. Moreover, one of the users (P18) of other editors mentioned that: *"Now that I know I am being tested for the usability of the tool, I pay more attention to it and I wonder why it [the interface design] is like this. I feel no one asked for users' opinion when designing it."*.

6.3.1.5 RQ5: How well does our Focus+Context Transition Editor alleviate Context-related challenges?

We used self-reported metrics to assess the Context-related challenges that the subjects faced when using the model editors. As mentioned earlier, we used three questions to measure the extent to which each subject experienced Context-related challenges (i.e., Q1, Q2, and Q3 in Fig. 6.1). Recall that Q1 asked about the challenges of relying on memory, Q2 asked about the challenges of switching focus diagrams, and Q3 asked about challenges of knowing about the relationships between artifacts.

The results of the subjects' answers to the Context-related challenges are presented in Table 6.9. Also, the average of the answers to each question for the subjects who used our editor versus the subjects who used other editors is depicted in Fig. 6.9. As can be

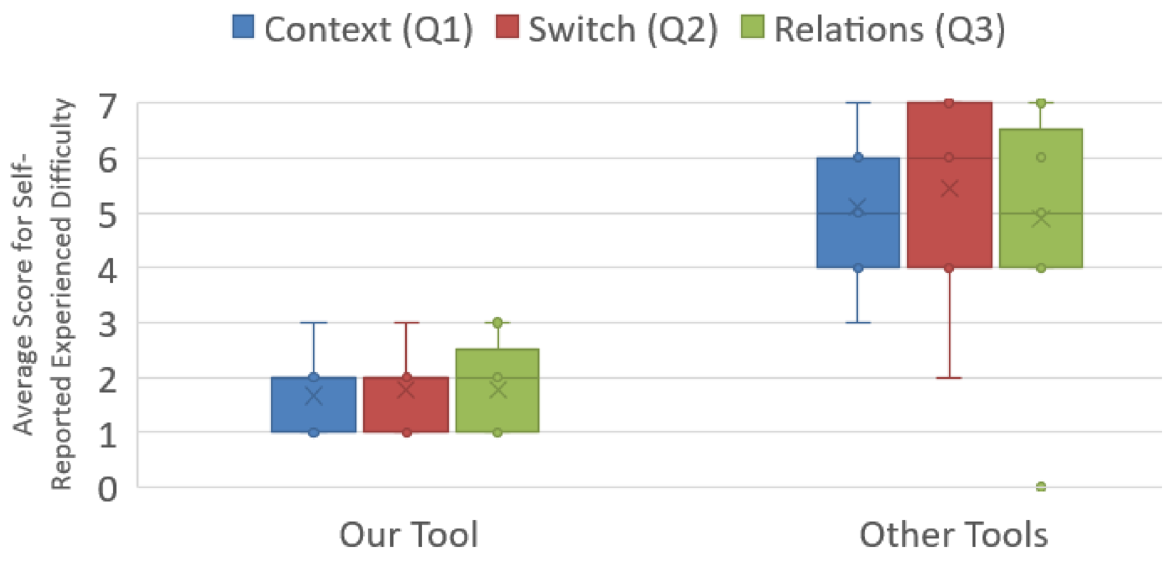


Figure 6.9: Context Study: Context-related challenges experienced on average by users of our editor vs. users of other editors.

seen, Context-related challenges were deemed to be experienced more than twice as much by users of other editors than by users of our editor.

The results of the lostness scores can also lead to the same conclusion about the users' experienced level of Context-related challenges, because one of the indicators of experiencing Context-related challenges is the number of switches to other diagrams. More specifically, in all of the tasks, the user needed to switch to the Class diagram to recall the contextual information that was relevant to performing a task. By dividing the total number of Class diagram views for all the tasks by 9 (number of tasks in the study), we can find that the average number of views for the Class diagram is equal to $(126/9) = 14$ for the users who used other editors and $(13/9) = 1.44$ for the users of our tool. This shows a significant improvement in the number of context switches between diagrams, which can indicate a significant improvement in alleviating the Context-related challenges.

The results of the analysis of the users' verbal statements also confirm the above results. We found that most of the users who used other editors expressed their experience of Context-related challenges during the sessions. For instance, one of the users (P18) expressed his Context-related challenges by saying: *"I feel I am more busy with remembering the elements than modelling itself."*. On the contrary, the users who used our editor did not make such strong dissatisfaction statements. Rather, they made positive statements about the Focus+Context Transition Editor. For example, a user (P11) said: *"It's good we have search [feature] here"*.

Table 6.12: Summary of the results of verbal analysis for the Context study.

	Overall		**Context**		**Reuse**		**Satisfaction**		**Syntax**	
Subj.	**Negative**	**Positive**	**Neg.**	**Pos.**	**Neg.**	**Pos.**	**Neg.**	**Pos.**	**Neg.**	**Pos.**
P1	7	1	4	0	1	1	0	0	2	0
P2	4	1	4	0	0	1	0	0	0	0
P3	2	0	2	0	0	0	0	0	0	0
P4	7	0	2	0	5	0	0	0	0	0
P5*	0	1	0	0	0	1	0	0	0	0
P6	5	4	1	0	0	1	3	2	1	1
P7*	4	5	1	1	0	1	1	3	2	0
P8*	3	2	0	0	0	0	2	2	1	0
P9	4	0	3	0	1	0	0	0	0	0
P10*	1	4	1	1	0	0	0	2	0	1
P11*	4	4	1	0	0	1	2	3	1	0
P12*	0	1	0	0	0	0	0	1	0	0
P13*	0	2	0	0	0	1	0	1	0	0
P14*	1	4	0	0	0	1	0	3	1	0
P15*	1	4	1	1	0	1	0	2	0	0
P16	6	3	2	1	0	0	2	2	2	0
P17	6	1	2	0	1	0	1	1	2	0
P18	4	1	1	0	0	1	3	0	0	0
Total No. of Comments on Our Tool	14	26	4	3	0	6	5	16	5	1
Total No. of Comments on Other Tools	45	12	21	1	8	4	9	6	7	1
No. of Subjects who Commented on Our Tool	6	8	4	3	0	6	3	7	4	1
No. of Subjects who Commented on Other Tools	9	7	9	1	4	4	4	4	4	1

* Denotes the subjects who used our tool.

6.3.1.6 Summary of Verbal Expressions

Table 6.12 shows a summary of the results of the verbal analysis performed on the subjects' verbal expressions for the Context study. As can be seen, we coded the expressions into four categories:

- *Context*: Remembering contextual information. An example of a positive comment is *"isRestStartTime [referring to the element in the Class diagram]? What was that?"*, and an example of a negative comment is *"[While the subject was saying:] What was the name of . . . O! Yeah, here it is!"*.
- *Reuse*: Reusing parts of the model (e.g., using parts of an existing transition in a new transition). An example of a negative comment is *"I'm trying to mimic the behaviour of one of the gates ... [the subject is struggling to take the State-Machine of one of the gates and reuse it for Gate E]"* and an example of a positive comment is *"[on task 6 which is related to reusing transition segments] Yeah! That was pretty easy!"*.
- *Satisfaction*: A general statement about the quality of the tool or a user's experience of using the tool. A negative comment could be *"Too complex..."*, and positive comment could be *"It was pretty easy."* or *"Nice"*.
- *Syntax*: A comment on the challenges of remembering the syntax of the language. An instance of a negative comment is *"Was it double equal or [single equal] . . . ? [while writing the transition guard]"* and an instance of a positive comments is *"It automatically started to add A1 [referring to the tool's feature in automatically inserting the grammar tokens in the text box]"*.

The table also shows the total number of negative or positive comments in each category and the number of users who made negative or positive comments for the tools.

Please note that the results of the subjects' total number of negative and positive comments on the tools should not be directly compared against each other because, in reality, some subjects may make more comments only because they tend to talk more than other subjects. Another reason to be cautious in drawing conclusions from the number of positive comments is that it is possible that the subjects' positive comments on our tool are biased due to the presence of the researcher (who is the developer of the tool). However, from the results we find that the number of subjects who made negative comments about our tool is fewer than the number of subjects who made negative comments about other tools for all the categories. In contrast, the number of subjects who made positive comments about our tool is greater than the number of subjects who made positive comments about other tools for all the categories.

Note also that the users of our tool made no negative comments with respect to *Reuse*, whereas users of other tools made eight negative comments in this category. We believe this is because of the reuse feature that we embedded in our Focus+Context Transition Editor which helps the user in Task 6. Also, four users of our tool made negative comments about *Context*, but there are nine users who made negative comments for the *Context* when using other tools. These observations support the results obtained for *RQ5*.

To support the results shown for *RQ4*, Table 6.12 shows that three users of our tool made negative comments about *satisfaction*, which is fewer than the number of users of other tools who made negative comments about *Satisfaction*. Moreover, the users of our tool expressed more positive comments about *Satisfaction* than the users of other tools.

6.3.1.7 Statistical Significance

The statistics related to Task Success, Errors on Tasks, Time on Tasks, Lostness, Users' Expectation versus Experience Ratings, and UMUX (Usability Metric for User Experience) questionnaire in the Context study are presented in Table 6.13. As shown by the *p values* in the table, values for subjects who used our tool were significantly different (*p values* $<= 0.05$) from the values for the subjects who used other tools. That is, our tool *significantly* outperformed other tools with respect to all the hypotheses in the Context study. We also calculate the effect size r based on the guidelines given by Andy Field [31] for Mann-Whitney's U test. The r values suggest a *high practical significance* ($r >= 0.5$) between the subjects who used our tool and the subjects who used other tools.

6.3.2 Results of the Debugging User Study

Our Interactive Consistency Management interface was evaluated with respect to its effectiveness in reducing the number of errors in subjects' models and in mitigating against the disruptive impacts of real-time error resolution.

Table 6.13: Statistics related to analysed metrics in the Context user study.

Collected Metric	**Tool**	**Mean**	**Median**	**Min**	**Max**	**Std. Dev.**	***p-value***	***r***
Task Success	Our Tool	0.99	1.00	0.94	1.00	0.02	**0.0032**	**0.70**
	Other Tools	0.85	0.89	0.44	1.00	0.17		
Task Errors	Our Tool	0.00	0.00	0.00	0.00	0.00	**0.0001**	**0.90**
	Other Tools	0.81	0.89	0.22	1.22	0.37		
Task Time	Our Tool	3.91	3.73	2.65	6.12	1.10	**0.0171**	**0.57**
	Other Tools	5.17	5.20	3.92	6.45	0.68		
Lostness	Our Tool	0.01	0.00	0.00	0.06	0.02	**0.0003**	**0.85**
	Other Tools	0.60	0.61	0.23	0.69	0.14		
Expectation vs. Experience Ratings	Our Tool	-1.69	-1.78	-2.33	-0.78	0.48	**0.0004**	**0.84**
	Other Tools	1.31	1.33	0.22	2.11	0.53		
UMUX	Our Tool	5.94	6.00	5.00	7.00	0.58	**0.0022**	**0.68**
	Other Tools	3.28	3.00	2.00	6.50	1.50		

6.3.2.1 RQ6: Are users of our Interactive Consistency Management interface more effective (i.e., create more correct models successfully) versus users of other editors?

The main research question in developing the Interactive Consistency Management interface was to assess its effectiveness in reducing the number of errors in subjects' models. To answer this question, we collected the task success scores and the number of errors in subjects' final models for each task and we compared the results for subjects using our editor against the results for the subjects using other editors. Our analysis shows that the subjects who used our editor were more successful in resolving inconsistencies and delivering more precise solutions than the subjects who used our editor.

Task Success: As shown in Fig. 6.10, participants who used our editor were more successful in locating and resolving inconsistencies than the participants who used other editors. Furthermore, the average success score on Task 4 dramatically dropped for the users who used other editors. This can be due to the fact that in Task 4 the subjects had to deal with more details about an error in order to correctly fix it. Specifically, in addition to detecting the error in the transition expression, the users needed to determine whether there were errors in the event, guard, or action segments of the transition. Thus, further investigations were required from the users to be able to resolve the error.

One can notice a significant difference in the task success scores between the two groups of users. The main reason is that other editors sacrifice correctness-by-construction in favour of usability or to avoid being intrusive. In our tool, we not only employed a correct-by-construction approach, but also implemented user-centric interfaces that could improve the users' experience of developing precise models. This indicates that most of the users' challenges can be alleviated with a well-designed and user-centered interface.

Table 6.14: Debugging Study: Results of each subject's success in completing each task.

Participant	Task1	Task2	Task3	Task4	Task5	Task6	Task7	Task8
P1	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0
P2	1.0	0.5	1.0	0.5	1.0	1.0	0.0	1.0
P3	1.0	0.5	1.0	0.5	0.5	1.0	0.5	0.0
P4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P5 *	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P6	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P7 *	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P8 *	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P10 *	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P11 *	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P12 *	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P13 *	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P14 *	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P15 *	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
P16	1.0	0.5	1.0	0.5	1.0	1.0	1.0	1.0
P17	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0
P18	1.0	1.0	0.5	1.0	0.0	1.0	1.0	1.0
mean	1.00	0.83	0.94	0.61	0.83	1.00	0.83	0.89
median	1.00	1.00	1.00	0.50	1.00	1.00	1.00	1.00
stdev	0.00	0.25	0.17	0.42	0.35	0.00	0.35	0.33
mean*	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
median*	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
stdev*	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

* Denotes the subjects who used our tool.

Table 6.15: Debugging Study: Result of the subjects' errors on each task.

Participant	Task1	Task2	Task3	Task4	Task5	Task6	Task7	Task8
P1	0	0	0	2	0	0	0	2
P2	0	1	0	0	1	0	2	0
P3	0	0	0	0	0	0	1	2
P4	1	0	0	0	0	1	0	0
P5 *	0	0	0	0	0	0	0	0
P6	0	0	0	0	0	0	0	1
P7 *	0	0	0	0	0	0	0	0
P8 *	0	0	0	0	0	0	0	0
P9	0	0	0	0	0	0	0	0
P10 *	0	0	0	0	0	0	0	0
P11 *	0	0	0	0	0	0	0	0
P12 *	0	0	0	0	0	0	0	0
P13 *	0	0	0	0	0	0	0	0
P14 *	0	0	0	0	0	0	0	0
P15 *	0	0	0	0	0	0	0	0
P16	0	0	0	0	5	2	0	0
P17	1	0	0	2	0	0	0	0
P18	0	0	1	0	8	0	0	1
mean	0.22	0.11	0.11	0.44	1.56	0.33	0.33	0.67
median	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
stdev	0.44	0.33	0.33	0.88	2.92	0.71	0.71	0.87
mean*	0	0	0	0	0	0	0	0
median*	0	0	0	0	0	0	0	0
stdev*	0	0	0	0	0	0	0	0

* Denotes the subjects who used our tool.

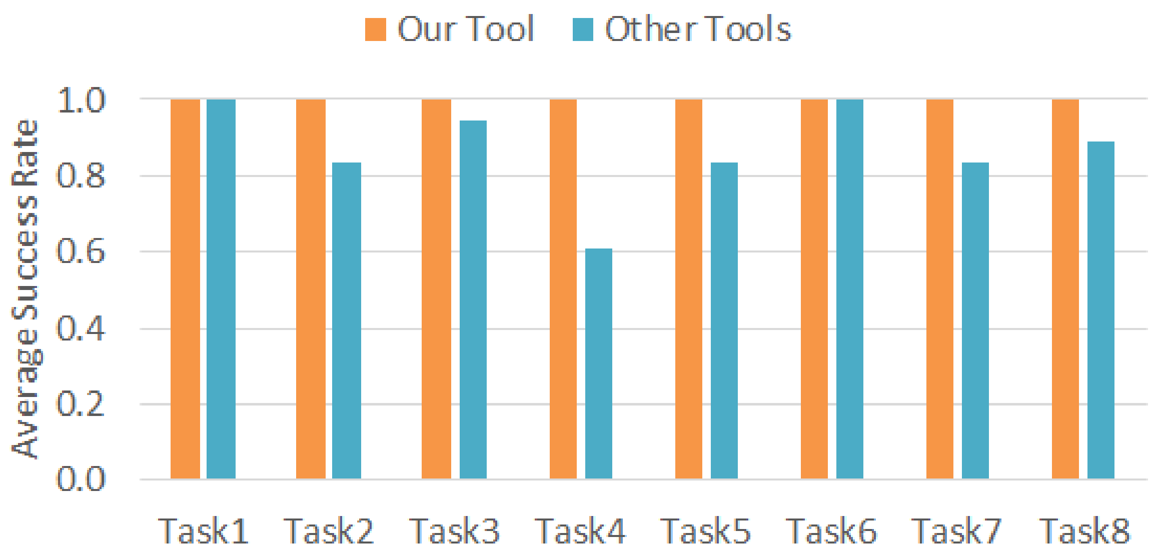


Figure 6.10: Debugging Study: Average success rate per task for users of our editor vs. users of other editors.

Errors on Tasks: As shown in Fig. 6.11, the subjects who used our editor submitted debugged models that had no errors (in any task), which is a significant improvement over the performance of the subjects who used other tools. The subjects who used other tools showed a low error rate for the first three tasks because these tasks were mostly related to renaming an element in the Class diagram and fixing the usages of the element in other diagrams. Despite the fact that such modifications are meant to be extremely easy, we can still observe that some users made errors especially when tasks involved larger models. Subjects committed or missed the most number of errors when performing Task 5, which involved type-mismatch errors. Most tools treat expressions (e.g., transitions' guard expression) as a plain-text field and perform no type-checking. Therefore, when a modeller changes the type of an element, most model editors cannot detect and report the resulting type error.

The zero number of errors for the subjects who used our tool is a *side-effect* of employing a correct-by-construction approach. We also found that collecting all of the inconsistencies into one interface, displaying the errors to the user, and allowing the users to resolve the errors in the same interface saves the users a lot of time and effort. However, this means that the user cannot move on with the task unless they fix the errors, which may not be desirable for many modellers who prefer a more relaxed consistency management approach. Also, most of our users discussed the necessity of tooling features (e.g., *apply to all*) that

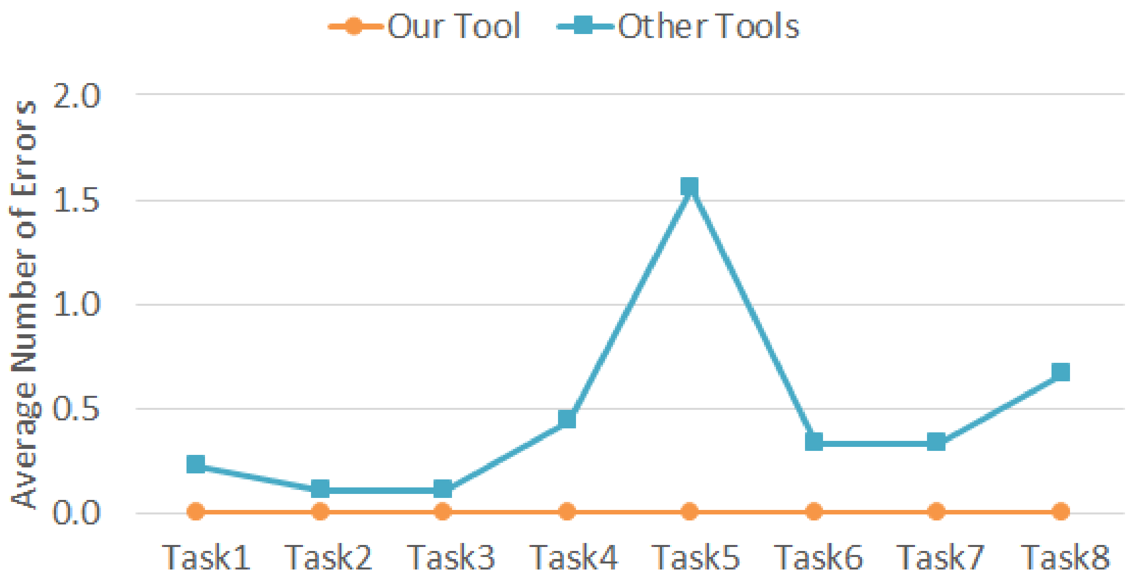


Figure 6.11: Debugging Study: Average number of errors per task made by users of our editor vs. users of other editors.

can handle the error-resolutions in larger models. Later, we show that the correct-by-construction approaches can be enhanced to not only reduce the number of errors, but also maintain the performance of the users. This is against current widely-spread claims in the literature, which emphasize that correct-by-construction approaches may interrupt and distract the user from their original tasks [3, 8, 9, 13, 27, 39].

Table 6.16: Debugging Study: Results of the subjects' time on each task (in minutes).

Participant	Task1	Task2	Task3	Task4	Task5	Task6	Task7	Task8
P1	02:30	02:25	01:20	06:04	03:30	02:49	03:57	03:54
P2	04:41	07:35	02:45	09:52	04:18	03:00	03:33	03:07
P3	03:32	04:40	05:38	03:18	02:47	02:24	03:41	03:40
P4	02:52	02:13	03:45	05:54	03:13	01:38	03:34	03:01
P5 *	01:44	01:38	00:40	02:35	04:11	01:06	03:14	01:38
P6	03:00	02:10	02:43	02:43	04:02	02:00	02:39	02:18
P7 *	01:35	02:42	00:49	02:15	06:13	01:59	01:59	01:50
P8 *	01:52	03:56	00:47	02:10	04:44	01:12	01:49	01:58
P9	02:50	02:37	03:57	03:05	02:35	01:10	02:22	02:18
P10 *	01:58	03:06	01:08	03:33	04:47	01:56	03:34	01:55
P11 *	01:35	03:12	01:19	02:11	05:48	01:34	03:08	02:40
P12 *	00:58	02:00	00:42	03:29	03:25	01:00	02:22	01:24
P13 *	01:49	03:28	00:44	02:47	04:29	01:21	03:07	02:01
P14 *	01:12	02:13	00:37	02:44	04:31	01:21	03:20	01:47
P15 *	00:56	02:36	00:37	03:09	03:39	01:11	01:56	02:21
P16	02:24	04:59	01:55	09:01	03:12	02:44	03:00	03:42
P17	03:52	03:02	04:08	03:53	03:19	03:14	02:30	02:38
P18	04:52	02:32	04:11	03:20	02:28	02:26	02:34	04:10
geo-mean	03:18	03:15	03:07	04:43	03:13	02:17	03:02	03:08
median	03:00	02:40	01:19	03:14	03:51	01:47	03:04	02:20
stdev	00:55	01:25	01:36	02:16	01:02	00:43	00:39	00:50
geo-mean*	01:28	02:40	00:48	02:43	04:34	01:22	02:38	01:55
median*	01:35	02:42	00:44	02:44	04:31	01:21	03:07	01:55
stdev*	00:23	00:44	00:15	00:32	00:54	00:21	00:41	00:22

* Denotes the subjects who used our tool.

Table 6.17: Debugging Study: Lostness results for the tasks for 18 subjects.

Participant	Task1		Task2		Task3		Task4		Task5		Task6		Task7		Task8	
	R,N,S	L	R,N,S	L	R,N,S	L	R,N,S	L	R,N,S	L	R,N,S	L	R,N,S	L	R,N,S	L
P1	1,5,5	0.80	1,2,4	0.71	1,3,3	0.67	1,2,6	0.83	1,5,5	0.80	1,5,5	0.80	1,5,10	0.94	1,5,8	0.88
P2	1,5,5	0.80	1,4,10	0.96	1,5,10	0.94	1,5,14	1.03	1,5,14	1.03	1,5,6	0.82	1,5,6	0.82	1,5,10	0.94
P3	1,5,6	0.82	1,5,5	0.80	1,5,7	0.85	1,5,5	0.80	1,5,5	0.80	1,5,5	0.80	1,5,5	0.80	1,5,5	0.80
P4	1,5,7	0.85	1,2,2	0.50	1,5,5	0.80	1,5,14	1.03	1,5,6	0.82	1,3,3	0.67	1,3,9	0.94	1,3,3	0.67
P5 *	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00
P6	1,5,5	0.80	1,5,5	0.80	1,5,5	0.80	1,5,5	0.80	1,5,8	0.88	1,5,5	0.80	1,5,7	0.85	1,5,5	0.80
P7 *	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00
P8 *	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00
P9	1,5,7	0.85	1,2,2	0.50	1,5,5	0.80	1,5,5	0.80	1,5,5	0.80	1,3,3	0.67	1,2,4	0.71	1,3,3	0.67
P10 *	1,1,1	0.00	1,1,1	0.00	1,2,2	0.50	1,2,2	0.50	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00
P11 *	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00
P12 *	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00
P13 *	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00
P14 *	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00
P15 *	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00	1,1,1	0.00
P16	1,1,1	0.00	1,5,5	0.80	1,5,5	0.80	1,4,4	0.75	1,5,5	0.80	1,5,5	0.80	1,5,6	0.82	1,5,8	0.88
P17	1,5,5	0.80	1,5,5	0.80	1,5,5	0.80	1,4,4	0.75	1,5,8	0.88	1,5,6	0.82	1,5,7	0.85	1,5,7	0.85
P18	1,5,5	0.80	1,5,5	0.80	1,5,5	0.80	1,5,7	0.85	1,5,5	0.80	1,5,5	0.80	1,5,5	0.80	1,5,8	0.88
mean		0.72		0.74		0.81		0.85		0.85		0.77		0.84		0.82
median		0.80		0.80		0.80		0.80		0.80		0.80		0.82		0.85
stdev		0.27		0.15		0.07		0.10		0.07		0.06		0.07		0.10
efficient subj. (< 0.5)		11.11%		0.00%		0.00%		0.00%		0.00%		0.00%		0.00%		0.00%
inefficient subj. (>= 0.5)		88.89%		100.00%		100.00%		100.00%		100.00%		100.00%		100.00%		100.00%
mean*		0.00		0.00		0.06		0.06		0.00		0.00		0.00		0.00
median*		0.00		0.00		0.00		0.00		0.00		0.00		0.00		0.00
stdev*		0.00		0.00		0.17		0.17		0.00		0.00		0.00		0.00
efficient subj. (< 0.5)*		100.00%		100.00%		88.89%		88.89%		100.00%		100.00%		100.00%		100.00%
inefficient subj. (>= 0.5)*		0.00%		0.00%		11.11%		11.11%		0.00%		0.00%		0.00%		0.00%

* Denotes the subjects who used our tool.

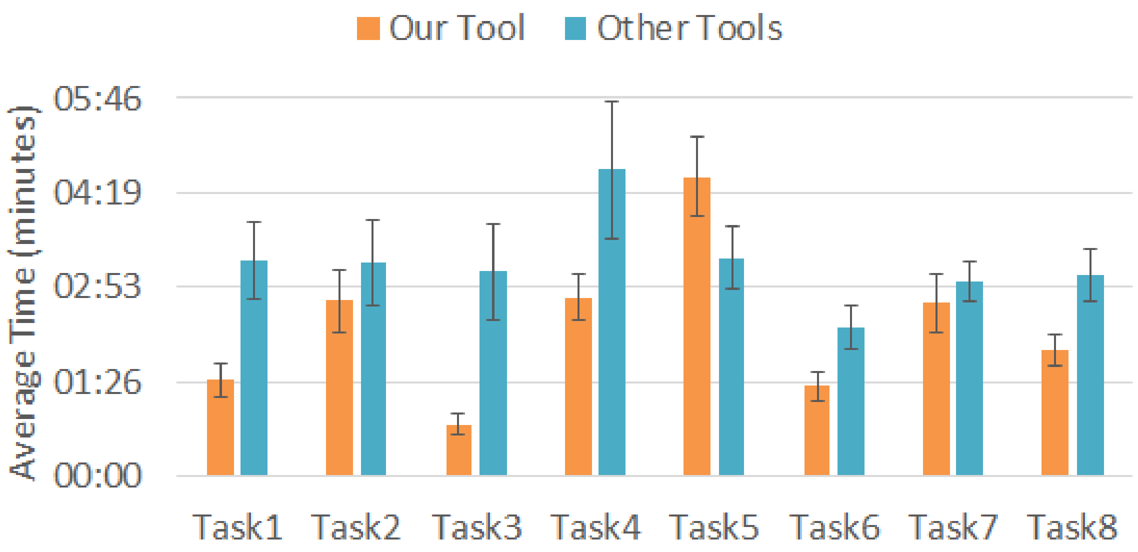


Figure 6.12: Debugging Study: Average time per task for users of our editor vs. users of other editors.

6.3.2.2 RQ7: Are users of our Interactive Consistency Management interface more efficient (i.e., spend less time on tasks) in creating precise models versus users of other editors?

Time on Tasks: It is not enough to be effective, if users are not also efficient. As shown in Fig. 6.12, subjects who used our editor completed all tasks, except Task 5, in less time on average than the subjects who used other editors. After reviewing the recorded video and audio files of the sessions, we noticed that the time on Task 5 was lengthened mostly because the users found it necessary to pause and provide feedback on how to improve the scalability of our interface. For instance, most of the subjects agreed that it would be good if the tool could provide an *“Apply to all”* mechanism (to support scalability) that could fix similar types of inconsistencies at once (e.g., writing a new guard expression for the first inconsistency and apply it to the rest of the inconsistent guards).

Lostness: Fig. 6.13 shows the average “lostness” scores per task for each group of subjects. As can be seen, all of the subjects who used other tools viewed many (irrelevant) diagrams, whereas the subjects who used our tool viewed only the diagrams that were specific to that particular task. The main reason was that our Interactive Consistency Management interface incorporated all the inconsistent elements in one view, which al-

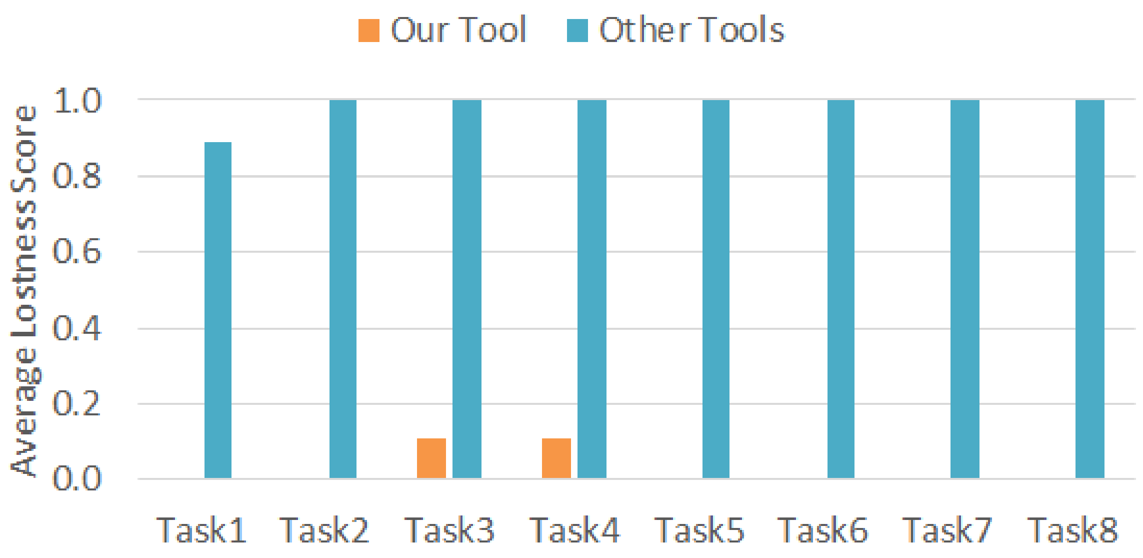


Figure 6.13: Debugging Study: Average lostness score per task for users of our editor vs. users of other editors.

lowed the subjects to fix the inconsistencies on the fly without needing to switch among diagrams. The results of the subjects' lostness scores is shown in Table 6.17. The table shows the percentage of *lost* subjects (i.e., inefficient subjects) who used other tools versus the percentage of *lost* subjects who used our tool. We set the R value (i.e., minimum number of diagrams to be viewed for a task) to 1 for all the tasks because in our opinion, a user should only open the intended diagram (e.g., Class diagram) to edit the model (e.g., renaming an element). It is the tool's responsibility to propagate the changes to the other parts of the model and inform the modeller about the changes that were made. One can oppose to our view and discuss that the R value should be 1+*the number of diagrams that referred to the edited element* simply because the user may double check the affected diagrams no matter what the tool does (e.g., because of scepticism).

6.3.2.3 RQ8: Are users of our Interactive Consistency Management interface more confident in the correctness of their models (i.e., solutions) versus users of other editors?

As shown in Fig. 6.14, the results of subjects' self-declared level of confidence indicate that the subjects who used our editor were more confident that they had addressed all

Table 6.18: Debugging Study: Post-session experience ratings.

Participant	Satisfaction (Q1)	Confidence	Capability (UMUX.Q1)	Usability (UMUX.Q2)
P1	5	5	5	5
P2	2	2	2	2
P3	2	2	2	1
P4	1	2	2	2
P5 *	6	6	7	7
P6	2	2	1	2
P7 *	7	7	7	6
P8 *	6	6	6	5
P9	1	5	2	3
P10 *	7	6	6	5
P11 *	6	6	6	7
P12 *	6	6	6	7
P13 *	6	7	6	6
P14 *	7	5	6	7
P15 *	6	7	6	6
P16	2	3	2	2
P17	2	2	1	2
P18	1	1	1	2
mean	2.00	2.67	2.00	2.33
median	2.00	2.00	2.00	2.00
stdev	1.22	1.41	1.22	1.12
mean*	6.33	6.22	6.22	6.22
median*	6.00	6.00	6.00	6.00
stdev*	0.50	0.67	0.44	0.83

* Denotes the subjects who used our tool.

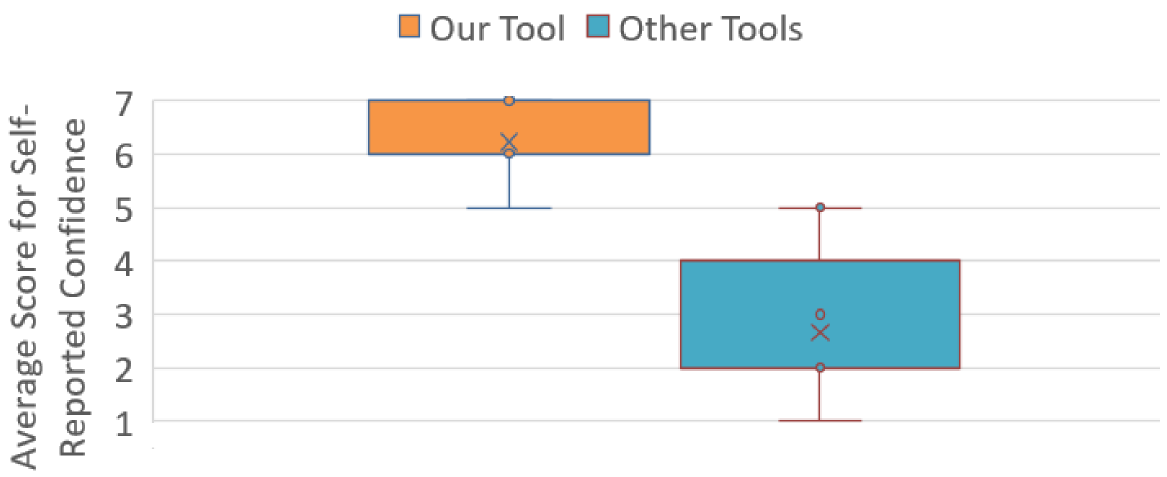


Figure 6.14: Debugging Study: Box-plot for the level of confidence for subjects using our editor vs. subjects using other editors.

the inconsistencies in the debugging tasks than the subjects who used other editors. One reason might be the improved error-resolution techniques that are provided by our tool. Most of the users who used our editor verbally expressed their satisfaction. For instance one of the users (P14) mentioned that: *"I do agree that it is nice that the tool tells me everything."*. However, there was one user of our tool (P10) who doubted his answer when we asked him whether he is confident with locating and fixing all the errors in the model. In fact, he said: *"I don't have trust in tools."*. Another subject (P14) also mentioned that *"I never trust in automatic [bug fixing]"*. Such a sceptical view from users about modelling tools is perhaps affected by his previous experiences with using the tools, and may be one of the obstacles to the adoption of MDE among practitioners. Such scepticism was also observed by Mussbacher et al. [71]. They illustrate that one of the barriers to the adoption of MDE is indeed the *not-so-good* past experiences of users with the tools: *"Maybe, the bad experience with CASE tools decades ago still casts a dark shadow on MDE"*. [71].

Additionally, we found that using a formal language or frameworks such as Xtext [26] that help in creating grammar-based models can also improve users' confidence in the correctness of their models, as stated by a user of our tool (P14): *"At least it will show errors related to syntax [because it uses a formal language]."*. This might be because of two reasons: 1) formal models are generally correct-by-construction models (at least syntactically), and 2) Eclipse-based frameworks (e.g., Xtext, GEF) guide the user in the process of model development by providing features such as Content-Assist that propose

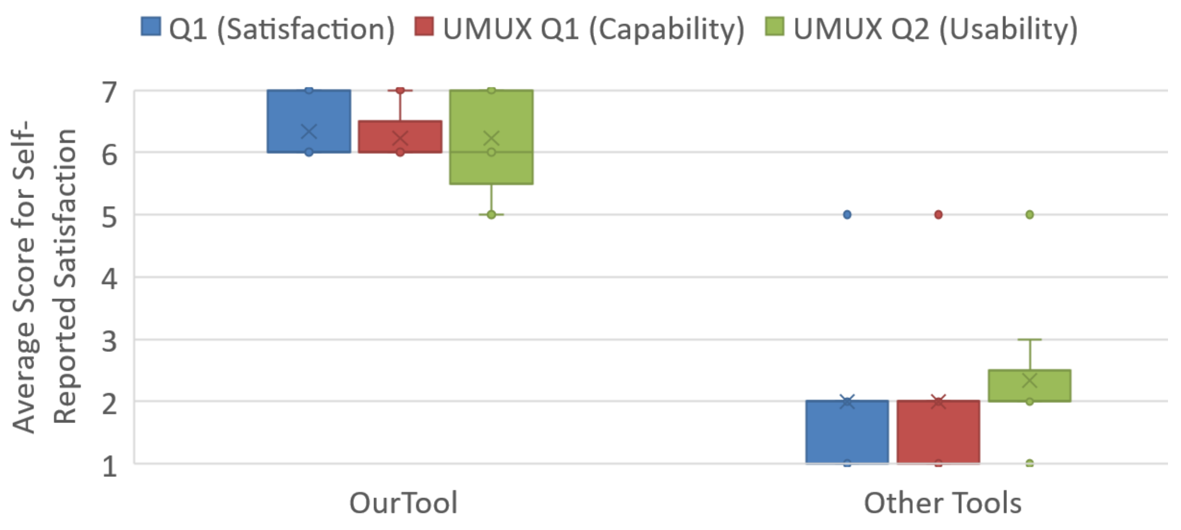


Figure 6.15: Debugging Study: Box-plot for the level of satisfaction of subjects who used our editor vs. subjects who used other editors.

the valid next steps to the user. This finding can also be confirmed by doing a comparison between the average confidence (3.33) of the users (i.e., P1, P2, P16) who used Capella (as a tool the also uses the Eclipse-based frameworks) and the average confidence (2.33) of the users who used other tools.

According to analysis of the verbal expressions of the users, the users who used other tools were less confident in their solutions, mostly because finding errors was a manual activity. One of the users (P17) stated that: *"I hope I got them [the errors], but how can I be sure if no auto-checking is done by the tool."*. Even for the tools that provided error detection, the users were not confident in their solutions. This is mostly because the tools did not provide sufficient interaction with the users (e.g., vague error messages, and not asking for acknowledgements from the user before or after applying a fix), which keeps the users out of the loop and uninformed.

6.3.2.4 RQ9: Does our Interactive Consistency Management interface provide a better level of user satisfaction versus other editors?

Users' satisfaction scores were collected by asking three questions (see Q1, UMUX.Q1, and UMUX.Q2 in Fig. 6.2). Q1 asked the users to rate whether their experience of using the tool was beyond their initial expectation or below. The UMUX.Q1 and UMUX.Q2 also

Table 6.19: Debugging Study: Pre-task expectation ratings.

Participant	Task1	Task2	Task3	Task4	Task5	Task6	Task7	Task8
P1	1	3	1	3	3	3	4	2
P2	1	2	1	2	2	2	3	3
P3	1	1	1	1	1	1	1	3
P4	1	1	1	1	1	1	2	2
P5 *	2	2	2	4	2	2	2	3
P6	1	2	1	2	2	2	2	2
P7 *	3	4	3	4	5	5	4	5
P8 *	4	4	4	4	4	4	4	5
P9	2	2	2	3	2	2	3	4
P10 *	4	3	4	6	4	4	7	5
P11 *	2	5	2	5	4	4	4	4
P12 *	5	5	4	5	6	6	6	6
P13 *	2	4	3	5	4	4	4	3
P14 *	3	2	3	6	3	2	4	4
P15 *	2	2	2	3	4	4	3	5
P16	1	2	1	3	2	2	2	4
P17	1	2	1	2	2	2	2	2
P18	1	2	1	2	2	2	2	2
mean	1.11	1.89	1.11	2.11	1.89	1.89	2.33	2.67
median	1.00	2.00	1.00	2.00	2.00	2.00	2.00	2.00
stdev	0.33	0.60	0.33	0.78	0.60	0.60	0.87	0.87
mean*	3.00	3.44	3.00	4.67	4.00	3.89	4.22	4.44
median*	3.00	4.00	3.00	5.00	4.00	4.00	4.00	5.00
stdev*	1.12	1.24	0.87	1.00	1.12	1.27	1.48	1.01

* Denotes the subjects who used our tool.

Table 6.20: Debugging Study: Post-task experience ratings.

Participant	Task1	Task2	Task3	Task4	Task5	Task6	Task7	Task8
P1	1	5	1	5	6	6	5	4
P2	4	4	1	4	6	7	7	7
P3	5	6	7	4	3	2	2	6
P4	2	1	3	4	2	2	3	3
P5 *	1	1	1	1	1	1	1	1
P6	3	4	5	4	5	4	4	4
P7 *	1	2	1	1	3	2	1	1
P8 *	1	2	1	3	3	1	2	2
P9	5	5	7	6	5	4	6	6
P10 *	2	1	2	4	2	1	2	1
P11 *	2	2	2	2	2	2	2	2
P12 *	1	2	1	2	3	1	2	1
P13 *	1	2	1	2	1	1	1	2
P14 *	1	1	1	1	2	1	1	1
P15 *	1	2	1	1	3	2	2	2
P16	2	5	5	4	5	5	6	7
P17	3	4	5	5	5	4	4	4
P18	4	4	4	4	5	5	4	4
mean	3.22	4.22	4.22	4.44	4.67	4.33	4.56	5.00
median	3.00	4.00	5.00	4.00	5.00	4.00	4.00	4.00
stdev	1.39	1.39	2.22	0.73	1.32	1.66	1.59	1.50
mean*	1.22	1.67	1.22	1.89	2.22	1.33	1.56	1.44
median*	1.00	2.00	1.00	2.00	2.00	1.00	2.00	1.00
stdev*	0.44	0.50	0.44	1.05	0.83	0.50	0.53	0.53

* Denotes the subjects who used our tool.

asked the users to rate how well the tool capabilities met their requirements and how easy-to-use the tool was, respectively. The results, shown in Fig. 6.15, demonstrate dramatically higher levels in subjects' satisfaction when using our editor. Furthermore, the results of the pre-task expectation ratings and post-task experience ratings for the subjects are shown in Tables 6.19 and 6.20, respectively. The tables show that while the users of other editors were disappointed with the level of model-debugging aids that were provided by their tools, the users of our editor were satisfied with the model-debugging support that they received from our editor. Again, as mentioned for the Context study, the users of other tools have lower pre-task assessments because the users of our tool took into account the effects of the unfamiliarity with our tool's debugging features. In fact, since *learnability* is one of the criteria for users' experience, we did not discourage them.

The results of the users' satisfaction for the Debugging study can be further confirmed by the analysis of the verbal statements expressed by the subjects. For example, one of the subjects (P1) who used other editors, expressed his dissatisfaction when performing Task 5 by saying: *"It is very difficult, it is not showing me where it is referenced, I have to do it manually, this is a pain ..."*. There were a few other participants who also expressed similar dissatisfaction statements during the tasks. Moreover, after finishing the task, we asked him if he thinks that he spotted *all* the errors in the diagrams. *"I wasn't sure. So, given more (larger) model I may need to look for everything ... not confident."*, he answered, and continued to the next task. This shows that most of the users expect and rely on the tools to provide proper consistency or correctness management features.

More specifically, the users who used other editors were more satisfied when performing tasks where changing an element in the Class diagram would affect the triggering event or action segments of transitions (that use the element), because their tools are generally able to trace the changed element to the transitions that use the element. However, the users were extremely disappointed when performing tasks where changing an element in the Class diagram would introduce an error in a guard segment of a transition, because guards are written as plain-text expressions that do not refer to actual model elements. Moreover, tools such as Capella that allow a guard expression to refer to actual model elements do not provide error recovery techniques. For instance, when an element is deleted from the Class diagram, Capella shows a dialogue box to the user indicating that the element is used in a transition, and deleting the element would cause the transition to be deleted. However, the user wanted only the affected segment (i.e., event, guard or action) of the transition to be deleted and not the whole transition. Therefore, the user had to find the affected transition, remove the segment that refers to the element to be deleted, and then return back to the Class diagram to delete the element. This mismatch between tool support and users' needs can cause a huge dissatisfaction in the users' view of the tools, and can lead to higher levels of disruption (discussed in *RQ10*).

Furthermore, although the subjects who used our tool were generally satisfied with our Interactive Consistency Management interface, they said that it would be good to have some "*apply to all*" feature, which can apply a fix to all instances of the same error type. In Task 5, a change in the model caused a number of type errors in the diagrams; one of the subjects (P11) mentioned that *"I wish there was a [auto apply] "*. Another user (P14) expressed his dissatisfaction multiple times during the session by saying *"It [the tool] could auto-pull things..."*. This indicates that, the subject were dissatisfied with the scalability of our error recovery techniques. One can conclude that, error-recovery techniques are as important as error-detection features when addressing inconsistencies.

Table 6.21: Debugging Study: Results of the disruption-related questions.

Participant	Expecting Disruption	Experienced Disruption (Q3)	Disruption-Worth Value (Q4)
P5	6	2	7
P7	4	1	7
P8	5	3	6
P10	7	2	7
P11	6	1	7
P12	6	1	6
P13	6	4	7
P14	6	1	7
P15	2	2	7
mean	5.33	1.89	6.78
median	6.00	2.00	7.00
stdev	1.50	1.05	0.44

6.3.2.5 RQ10: Do modellers who use our Interactive Consistency Management interface judge the interface as being too intrusive and disruptive to them given the increased level of correctness that it provides?

One of the main hypotheses underlying the design of our Interactive Consistency Management interface is that addressing inconsistencies in real-time while editing a model does not have to be intrusive to the modeller (in contrast to what researchers believe) if tool developers take human cognitive factors into account when designing interfaces. For the users of our editor, the results of their self-reported expected disruption asked in a pre-session questionnaire and their self-reported experienced disruption asked post-session is depicted in the box plot shown in Fig. 6.16. It shows that the subjects strongly expected that maintaining model consistency at all times would be intrusive. In fact, one of the subjects (i.e., P10) mentioned that “Oh! Yeah! It will be very intrusive.”. However, after using our editor, their experience ratings showed the opposite. That is, their mean experience rating indicates that they barely felt disrupted when asked by our editor to debug inconsistencies during model editing. The subject P10 post-session rating shows that he did not experience disruption during his tasks. This can also be confirmed by looking at the times on tasks for the users who used our tool, as a user’s disruption would possibly cause them to spend more time on the tasks but it was not the case for the users of our tool.

Furthermore, we asked the subjects to rate their views of whether experiencing disruption was worth delivering more correct models. Table 6.21 demonstrates the results of the disruption-related questions. It shows that subjects, on average, expected to be disrupted by the tools (mean value of 5.33), they experienced much less than what they expected (mean value of 1.89). However, the high mean value of their expectations might be because they were going to experience a new tool (i.e., our tool) and the general negative belief that the community has regarding the correct-by-construction approach. Also, the lower mean value of experience ratings can be because of the size of the model. That is, the same users might have rated differently if the model was larger and more complex. However, we can at least conclude that employing correct-by-construction techniques does not have to be intrusive to the user if proper tooling techniques are devised.

Moreover, for the tasks that required users’ action in resolving errors (e.g., Task 2 and Task 4), the users who used our editor obtained better lostness scores and times on the tasks versus the users who used other editors. These can be other indicators that users of our editor have not experienced any unusual intrusiveness when performing their tasks using our tool.

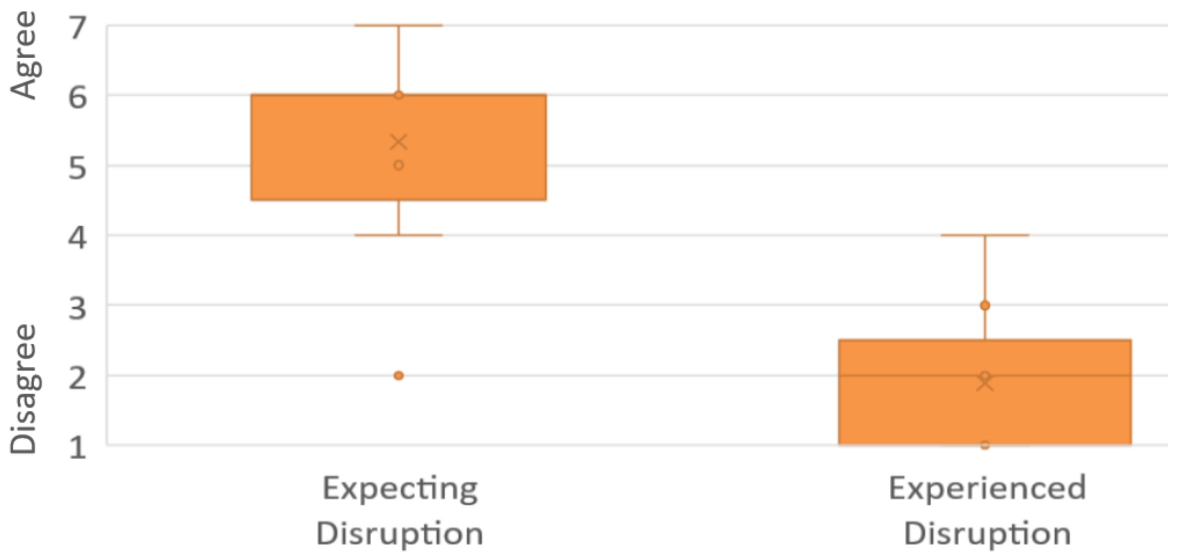


Figure 6.16: Debugging Study: Subjects' assessment of the intrusiveness of our Interactive Consistency Management interface.

Table 6.22: Summary of the results of verbal analysis for the Debugging study (part 1).

	Overall		**Context**		**Satisfaction**		**Syntax**		**Recognition**		**Intrusiveness**	
Subj.	**Negative**	**Positive**	**Neg.**	**Pos.**	**Neg.**	**Pos.**	**Neg.**	**Pos.**	**Neg.**	**Pos.**	**Neg.**	**Pos.**
P1	31	5	4	0	0	0	0	0	6	1	0	0
P2	9	6	0	1	0	0	0	0	1	0	0	0
P3	3	0	0	0	0	0	0	0	0	0	0	0
P4	5	0	1	0	1	0	0	0	0	0	0	0
P5*	1	0	0	0	0	0	0	0	0	0	0	0
P6	3	1	0	0	0	0	0	0	0	1	0	0
P7*	5	6	1	0	0	3	1	0	0	0	0	2
P8*	1	3	0	0	1	1	0	1	0	0	0	0
P9	1	1	0	0	0	0	0	0	0	1	0	0
P10*	6	8	1	0	1	8	0	0	0	0	0	0
P11*	5	3	0	0	0	2	0	0	0	0	0	0
P12*	2	0	0	0	0	0	0	0	0	0	0	0
P13*	1	1	0	0	0	1	0	0	0	0	0	0
P14*	8	5	0	1	0	0	0	0	0	2	0	1
P15*	1	3	0	1	1	1	0	0	0	0	0	0
P16	10	0	0	0	2	0	0	0	3	0	0	0
P17	4	0	0	0	0	0	0	0	1	0	0	0
P18	5	0	0	0	0	0	0	0	0	0	0	0
Total No. of Comments on Our Tool	28	29	2	2	3	16	1	1	0	2	0	3
Total No. of Comments on Other Tools	73	13	5	1	3	0	0	0	11	3	0	0
No. of Subjects who Commented on Our Tool	8	7	2	2	3	6	1	1	0	1	0	2
No. of Subjects who Commented on Other Tools	10	4	2	1	2	0	0	0	4	3	0	0

* Denotes the subjects who used our tool.

Table 6.23: Summary of the results of verbal analysis for the Debugging study (part 2).

	Association		**Action Planning**		**Confidence**		**Scalability**	
Subj.	**Negative**	**Positive**	**Neg.**	**Pos.**	**Neg.**	**Pos.**	**Neg.**	**Pos.**
P1	11	2	4	1	6	1	0	0
P2	3	2	4	3	1	0	0	0
P3	0	0	2	0	1	0	0	0
P4	1	0	1	0	1	0	0	0
P5*	0	0	1	0	0	0	0	0
P6	0	0	2	0	1	0	0	0
P7*	0	0	1	1	0	0	2	0
P8*	0	0	0	1	0	0	0	0
P9	0	0	0	0	0	0	1	0
P10*	0	0	0	0	2	0	2	0
P11*	0	0	2	2	0	0	2	0
P12*	0	0	1	0	0	0	1	0
P13*	0	0	0	0	1	0	0	0
P14*	0	1	3	0	2	0	3	0
P15*	0	0	0	1	0	0	0	0
P16	2	0	1	0	2	0	0	0
P17	1	0	1	0	1	0	0	0
P18	1	0	2	0	2	0	0	0
Total No. of Comments on Our Tool	0	1	7	5	5	0	9	0
Total No. of Comments on Other Tools	19	4	18	4	15	1	2	0
No. of Subjects who Commented on Our Tool	0	1	4	4	3	0	4	0
No. of Subjects who Commented on Other Tools	6	2	9	2	8	1	2	0

* Denotes the subjects who used our tool.

6.3.2.6 Summary of Verbal Expressions

Tables 6.22 and 6.23 show a summary of the results of the verbal analysis performed on the subjects' verbal expressions during the Debugging study. We coded the expressions into the following categories.

- *Context*: Remembering contextual information.
- *Satisfaction*: A general statement about the quality of the tool or user's experience of using the tool.
- *Syntax*: Remembering the syntax of the language.
- *Recognition*: Recognizing whether there are errors in the model or not. A negative comment could be *"Again, it is not saying if there is any problem, if there is any error."* and a positive comment could be *"It is highlighted."*.
- *Intrusiveness*: A statement that shows a user's opinion of feeling disrupted or not. An instance of a positive comment is *"Honestly, it was only for the necessary parts, it was not disturbing"*.
- *Association*: Locating errors that are introduced to the model as a result of editing or deleting an element. An example of negative comment is *"This is by no means an efficient way that I have to go through all diagrams."*, and an example of a positive comment is *"I do agree that it is nice the tool tells me everything [referring to the tool's feature to show the inconsistencies and their locations.]"*.
- *Action Planning*: Resolving an error (i.e., edit, delete, or delete parent element). An example of a negative comment is *"Didn't change [surprised that the tool did not automatically propagated the name change throughout the model]?"*, and an example of a positive comment is *"So it was completely automatic [referring to the auto-fix]."*.
- *Confidence*: A statement that indicates whether a user is confident with the correctness of the model after resolving errors. An instance of a negative comment is *"I'm not sure, there should be somewhere that it tells me all deleted items are here . . . "*, and an instance of a positive comment is *"It changed everything, no problem there."*.
- *Scalability*: A statement about the lack of scalability in our User-Interactive Consistency Management Interface or in other tools, in case there are too many inconsistencies. An instance of a negative comments about *Scalability* is *"Apply all or maybe copy and paste [may be required]"*.

The tables show the total number of negative or positive comments that fall in each category, and the number of users who made negative or positive comments for the tools. Again, because different users have different tendencies to speak their thoughts loud (i.e., think-out-loud), their total number of negative and positive comments on the tools may not be directly compared against each other. However, it can be seen that the subjects were more satisfied with our tool with respect to the four categories that our User-Interactive Consistency Management interface claims to improve (see 5.6.2): *Recognition, Association, Action Planning, and Confidence.* For *Recognition* and *Association*, the users of our tool made no negative comments, perhaps because our tool pops-up a dialogue box that lists all the errors and their location in the model (e.g., containing diagrams). For *Confidence*, three users of our tool made negative comments, which is less than the number of users of other tools who made negative comments (i.e., eight users). Two of the three subjects who made negative comments on our tool expressed that they never trust any tool. With respect to *Action Planning*, four users made negative comments when using our tool (which is still fewer than the number of users who made negative comments when using other tools). We found that these four users' negative comments about *Action Planning* were due to the lack of *Scalability* of our *Action Planning* techniques. As shown in Table 6.23, we received many comments from users suggesting that our tool may require a better scalability-related feature when there are many errors that need to be resolved. The subjects who used our tool commented that an *"Apply to all"* type of functionality would be helpful when fixing errors that need a common fix.

As shown in Table 6.22, the results of verbal statements regarding *Intrusiveness* supports our results for *RQ10*. That is, not only were there zero negative comments related to *Intrusiveness*, but also there were positive comments from two of the users who used our tool. The subjects who used other tools also made no negative comments, but this was expected because most of the tools do not enforce correct-by-construction. Only Capella shows a pop-up dialogue box that immediately notifies users about an error. Other tools such as Papyrus decorate an erroneous element in the model with an icon showing that the element is erroneous.

The results regarding *Satisfaction* indicate that subjects who used our tool made sixteen positive comments and three negative comments about our tool, which in general supports our results for *RQ9*. The three negative comments are mostly related to the fact that our tool showed less robustness (sometimes because of limitations in the underlying frameworks) compared to other tools which are designed for industrial uses.

6.3.3 Statistical Significance

The quantitative analysis of our results using Mann-Whitney U test [100] for the Debugging study with respect to the Task Success, Errors on Tasks, Time on Tasks, Lostness, Users' Expectation versus Experience Ratings, and UMUX (Usability Metric for User Experience) are presented in Table 6.24. As shown by the *p values* in the table, values for subjects who used our tool were significantly different (*p values* <= 0.05) from the values for the subjects who used other tools. That is, our tool *significantly* outperformed other tools with respect to all the hypotheses. Also, the effect size *r* indicates a *high practical significance (r* >= 0.5*)* between the subjects who used our tool and the subjects who used other tools.

Table 6.24: Statistics related to analysed metrics in the Debugging user study.

Collected Metric	**Tool**	**Mean**	**Median**	**Min**	**Max**	**Std. Dev.**	***p-value***	***r***
Task Success	Our Tool	1.00	1.00	1.00	1.00	0.00	**0.0051**	**0.68**
	Other Tools	0.87	0.88	0.63	1.00	0.13		
Task Errors	Our Tool	0.00	0.00	0.00	0.00	0.00	**0.0005**	**0.82**
	Other Tools	0.47	0.38	0.00	1.25	0.38		
Task Time	Our Tool	2.32	2.31	1.92	2.74	0.28	**0.0010**	**0.78**
	Other Tools	3.42	3.27	2.61	5.04	0.74		
Lostness	Our Tool	0.01	0.00	0.00	0.13	0.04	**0.0002**	**0.88**
	Other Tools	0.80	0.81	0.71	0.92	0.06		
Expectation vs. Experience Ratings	Our Tool	-2.26	-2.25	-3.75	-1.38	0.74	**0.0003**	**0.84**
	Other Tools	2.46	2.50	1.25	3.13	0.64		
UMUX	Our Tool	6.22	6.50	5.50	7.00	0.51	**0.0003**	**0.88**
	Other Tools	2.17	2.00	1.50	5.00	1.12		

6.4 Threats to Validity

The main threat to validity is the participants' competency with the UML and UML editors. We tried to mitigate against this threat by asking the subjects to self-declare their familiarity with the UML and editors. In addition, we embedded a UML exercise in our recruitment questionnaire and only recruited those subjects who could pass the exercise.

Another threat to external validity is the participants' familiarity with the application domain and the system's complexity. To address this threat, we chose a simple and familiar application domain. The size of the model in our studies is not comparable to the large-scale complex systems (e.g., see [56]), and it is possible that the performance improvements that the subjects demonstrated in the studies would not scale to much larger models. To cope with this threat, we designed the task to be simple enough so that the subjects could perform their tasks without a complete understanding of the system. Moreover, we designed Task 9 in the Context study and Tasks 4 and 5 in the Debugging study to be more difficult than other tasks, so that we can observe how the users' performance would be affected when performing more difficult tasks. The results show that there is still an improvement for the subjects who used our tool versus the subjects who used other tools.

The small number of subjects can also be another threat to validity, but the number is recognized as adequate by various sources [7][92]. Also, some of the subjects were repeated from the formative study. This might affect their familiarity with the application domain because they used the same application domain and Class diagram in the previous formative study. However, we believe that the effects of the repeated subjects is minimal because there was a one-year gap between the two studies, so it is unlikely that they could recall the details from the previous formative study. Also, we designed different sets of questions in the Context and Debugging studies than those that were used in the formative study to mitigate against the effects of the repeated subjects on the results. Furthermore, it is possible that the pre-task expectation ratings of the subjects who participated in our formative study were biased towards higher expectation ratings because the experienced related challenges in the formative study.

One of the threats to the construct validity of our studies is related to the design of pre-task and post-tasks ratings. We believe that a 7-scale rating is too wide and may produce results that are not quantitatively meaningful. That is, subjects may know that their expectation of a tool is high, but they may not know if it is e.g., 5, 6, or 7. We counter-attacked this threat during the interpretation of our results.

One can argue that the order of the tasks should be randomized. However, we designed the tasks such that they become more and more difficult from the first task to the last task to support the users' learning curve [14][78].

Another threat that may impact the validity of the conclusions drawn from users' verbal expressions is the possibility of subjects biases on making more positive comments and less negative comments on our tool due to the presence of the researcher (who is the developer of the tool). Also, we found that, in general, users tend to make negative comments rather than positive comments. That is, users spot weak points of the tools more than strengths of the tools.

6.5 Discussion

In this section, we suggest additional implications (conjectures) inferred from the subjects' behaviours and attitudes about the tools that were collected during the subjects' sessions with the tools. We think tool researchers and vendors can benefit from these implications and can undertake further investigations on these implications.

- The results suggest that modellers benefit from a sliced view (from related diagrams) of the context of their modelling task. Developing such interfaces should be encouraged among the tool vendors for all types of diagrams to minimize the users' memory load. Moreover, the significant gap between the experienced cognitive difficulties of the two groups of users suggests that Focus+Context editors can play a critical role in alleviating users' cognitive load- especially in reducing the challenges of remembering and editing contextual information.

- A user might be negligent in creating correct and precise models unless they are constrained by explicit guidelines and directions. It is the tool's responsibility to offer useful hints and notifications to refresh a modeller's memory about the explicit guidelines and best practices for improving the model's quality. One can also notice the necessity of collecting and developing a standardized and comprehensive list of such guidelines for modelling, which can be implemented and embedded in the model editors.

- We found that most of the users who used other tools complained about developing expressions in plain text or in text boxes that do not provide sufficient aids. Thus, the modelling tools should devise easy-to-use expression editors (as opposed to plain-text editors) that enable users to develop expressions that can be parsed and verified as being semantically sound. These expression editors should employ a rich grammar to cover all types of expressions.

- Auto-fixing errors is not enough. Users prefer to know about the steps that are being taken in the background. For instance, we found that users may want to be notified if a tool has automatically propagated a change (e.g., renaming an element). Therefore, the tools should seek for users acknowledgement before applying any auto-fixes (even though it might be clear to the user). Perhaps, a proficient tool can allow the user to modify such preferences in the tool's settings to accommodate different users' preferences. Moreover, users feel more confident when a tool warns them explicitly about errors in the model, even when the error is trivial and easy to locate.

- Resolving errors as they occur may or may not be intrusive and disruptive to users. The results suggest that users with sufficient tooling support do not feel disrupted by on-the-fly error resolution. Undertaking on-the-fly error resolution with extra tooling support, such as error recovery techniques, results in improving quality of models and reducing the impacts of disruption on users' performance.

6.6 Conclusion

In the previous chapter, we proposed tooling techniques and enhancements (i.e., automation and Focus+Context user interfaces) that aim to reduce efforts of *Context* and *Debugging* challenges when designing UML Class and State-Machine diagrams. In this chapter, we presented the results of the empirical user studies that we conducted to assess the effectiveness of our proposed tooling techniques on human users. The results of our studies indicate that employing a Focus+Context approach can significantly improve users' satisfaction of using model editors and can increase their effectiveness and efficiency in editing and debugging models. Our results have implications for tool vendors to enhance and improve the quality of UML model editors, which is crucial for a greater adoption of MDE by industry.

Chapter 7

Conclusion and Future Work

This chapter briefly reviews the thesis and its contributions, and suggests directions for future work.

7.1 Summary of Thesis and Contributions

In this thesis, we introduced our approach, referred to as **U**ser-**C**entric and **A**rtefact-Ce**n**tric **D**evelopment **o**f **Models** (*UCAnDoModels*), which employs and incorporates a balanced set of user-centric and artefact-centric techniques that, combined together, ease the tasks of editing and debugging models. We proposed that applying such user-centric techniques to the design and development of modelling tools would enhance the quality of UML-like modelling tools. Our approach consists of the following steps: 1) Analysing users and their tasks to identify the greatest challenges of using tools, 2) Identifying the cognitive factors that underlie each challenge, 3) Using standard practices or devising techniques to alleviate the identified challenges by addressing the underlying cognitive factors, and 4) Assessing the effectiveness of the proposed techniques on human users. This thesis provides valuable insights into how modellers interact with model editors and how cognition psychology theories can improve this interaction. The remainder of this section provides brief descriptions of the main contributions of the thesis.

7.1.1 Users and Tasks Analysis

As the first step of our thesis, we conducted a formative user study to understand modellers' challenges of using modelling tools. By observing the subjects in our user study, we identified ten tooling difficulties:

- *Order*: Performing tasks in the right order,
- *Context*: Remembering the relevant contextual information when performing a task,
- *Navigation*: Writing correct and precise navigation expressions,
- *Syntax*: Knowing the correct syntax of the language,
- *Type-Matching*: Writing expressions that are free of type errors,
- *Debugging*: Locating, understanding, and fixing errors in the model,
- *Layout*: Organizing the diagrams' layout,
- *Views*: Arranging different diagrams besides each other for a better view,
- *Reuse*: Reusing (i.e., copy and pasting) model elements, and
- *Look*: Setting appearance-related properties (e.g., shapes and colors).

Our analysis revealed that among all the identified difficulties, *Context* and *Debugging* challenges are the most severe. This may be because existing modelling tools take a language specification-based approach and implement their tools based mostly on the specification of the language. For instance, current tools provide separate diagrams for the structural diagrams (i.e., Class diagram) and the behavioural diagram (e.g., State-Machine diagram) because the UML specification supports separation of concerns, including the separation of the behavioural diagrams from their corresponding structural diagram (or elements). However, this separation puts a burden on the user to recollect the information that is separated within different diagrams. The results of our formative user study indicate that tool researchers and tool vendors should pay more heed to proposing interfaces and techniques that can mitigate or eliminate such difficulties.

7.1.2 UCAnDoModels Model Editor

We identified the underlying cognitive factors for the two aforementioned most-severe challenges: Short-term and Long-term Memory, Transience, Absentmindedness, Misattribution, Lack of Context, Execution Failures, and Planning Failures. Accordingly, we proposed automation features and several *Focus+Context* solutions that can alleviate the overall cognitive load of modellers by providing the contextual information that is relevant to performing a particular modelling task. We implemented a semi-formal (FORML-based) UML-like model editor as an Eclipse plugin.

7.1.3 Empirical Implications for Tool Researchers

We conducted two user studies to assess the impacts that user-centric interfaces, such as our *Focus+Context* interfaces, have on improving the users' experience of using model editors (and tools in general) with respect to the users' effectiveness, efficiency, satisfaction, etc. According to the results, employing user-centric interfaces can significantly contribute to enhancing the productivity and satisfaction of modellers and improving the quality of models, such as: 1) improving the users' ability to successfully fulfil their tasks, 2) avoiding unnecessary context switches among diagrams, 3) producing more error-free models, 4) remembering contextual information, and 5) reducing time on tasks. Moreover, we provided observation-based implications for the tool researchers that may eliminate many of the existing tooling challenges. However, further studies may be required to investigate some of these implications.

In summary, we have shown that: *current MDE tools can be enhanced by employing artefact- and user-centric techniques; the resulting tools help reduce the effort of editing and debugging analysable models and improve users' experience of using model editors by incorporating techniques that take into account human-cognitive factors.*

7.2 Limitations

Our dissertation has potential limitations, which are listed below.

- Narrow formulation of research objectives: We acknowledge that the research objectives (i.e., investigating in the context of Class and State-Machine diagrams) might not be broad enough to enable a complete generalization of the results. We think this can be considered as a potential future work of our thesis.

- Sample sizes of the user studies: One can argue that the sample sizes used in our user studies is small, which may prevent to identify significant relationships within the results. However, the number of 18 participants are determined to be sufficient when performing usability testing [7], even though conducting the studies with larger sample sizes could generate more accurate results.

- Subjects' incompetency: It is arguable that recruiting students may weaken the applicability of the results to outer population (i.e., industrial modellers in our case). However, we set several eligibility requirements to ensure that the subjects are competent enough to participate in the studies.

- Insufficient previous studies in the literature: We could find a little amount of research related to evaluating user experience in the context of UML modelling tools. This makes it difficult to 1) design studies that are perhaps more directed towards modelling tools (versus the general tools) and 2) compare our findings to the finding from other researchers.

- Imperfect design of the contextual interfaces: We believe that there is room for enhancing the proposed interfaces by performing more in-depth analysis on the design of the interfaces. Certainly several design decisions could be improved. For instance, different users (e.g., novice modeller or experienced modeller) have different preferences about the design of the interfaces and what should be viewed as the context. Thus, while a design may be desirable for a user, it might be rejected by another user. Overcoming this limitation requires further investigations that was beyond the scope of this thesis.

7.3 Future Work

As a possible future work, one can conduct a formative study to investigate the critical challenges of developing other types of models (e.g., UML Sequence diagrams, Simulink models). If the same challenges are identified (i.e., *Context* and *Debugging*), then it would be desirable to accept our tooling techniques to address the challenges and assess their generalizability on other types of models. Similarly, exploring the applicability of our proposed techniques to other model editors or even programming IDEs can be seen as a future work. By doing so, most of the burden that is put on the modellers or programmers will be alleviated by the modelling or programming tools.

Furthermore, our proposed solutions may be useful in the context of collaborative modelling, where the awareness of modellers about the model (i.e., contextual information) is reduced mostly because a model is developed by different modellers who may not have participated in the collaboration due to various reasons (e.g., geographical distances). Thus, one can extend the context of our work and investigate the effectiveness of our user-centric interfaces in the context of collaborative modelling.

As another future direction, a more thorough analysis of UML models and meta-model can be performed to exploit a more general, model-driven, and automatic approach for extracting contextual information relevant to any modelling task. Currently, we identified the sources of contextual information by observing the users in our formative user study, as well as analysing the meta-model of the FORML language. However, pulling *all* of the cross-cutting information into one interface may not work well due to the issues of scalability. A more intelligent approach would be to prioritize the contextual elements and display those information that are more likely to be what the user needs for completing their current modelling task.

Last but not least, it is possible that different results would be drawn in more sophisticated evaluations, such as recruiting industrial modellers as subjects, designing more complex tasks and larger models, and assessing more complex application domains. Therefore, a possible future work can be to reevaluate the interfaces in larger models and improve them to perform better in more sophisticated contexts.

References

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

[2] Silvia Abrahão, Kasper Hornbæk, Effie Law, and Jan Stage. Interplay between Usability Evaluation and Software Development (I-USED 2009). pages 969–970. Springer, Berlin, Heidelberg, 2009.

[3] Piotr D Adamczyk and Brian P Bailey. If not now, when?: the effects of interruption at different moments within task execution. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 271–278. ACM, 2004.

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

[5] Henning Agt-Rickauer, Ralf-Detlef Kutsche, and Harald Sack. Domore-a recommender system for domain modeling. In *MODELSWARD*, pages 71–82, 2018.

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

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

[8] Anja Baethge and Thomas Rigotti. Interruptions to workflow: Their relationship with irritation and satisfaction with performance, and the mediating roles of time pressure and mental demands. *Work & Stress*, 27(1):43–63, 2013.

[9] Brian P Bailey and Joseph A Konstan. On the need for attention-aware systems: Measuring effects of interruption on task performance, error rate, and affective state. *Computers in human behavior*, 22(4):685–708, 2006.

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

[11] Stefan Berlik. *Eclipse Modeling Framework*. Addison-Wesley, 2007.

[12] Xavier Blanc, Isabelle Mounier, Alix Mougenot, and Tom Mens. Detecting model inconsistency through operation-based model construction. In *30th International Conference on Software Engineering*, pages 511–520. IEEE, 2008.

[13] Deborah A Boehm-Davis and Roger Remington. Reducing the disruptive effects of interruption: A cognitive framework for analysing the costs and benefits of intervention strategies. *Accident Analysis & Prevention*, 41(5):1124–1129, 2009.

[14] Francis Bordeleau, Grischa Liebel, Alexander Raschke, Gerald Stieglbauer, and Matthias Tichy. Challenges and research directions for successfully applying mbe tools in practice. In *MODELS (Satellite Events)*, pages 338–343, 2017.

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

[16] Marcel Bruch, Thorsten Schäfer, and Mira Mezini. On evaluating recommender systems for API usages. In *Proceedings of the 2008 International Workshop on Recommendation Systems for Software Engineering*, RSSE '08, pages 16–20, New York, NY, USA, 2008. ACM.

[17] Raluca Budiu. Memory recognition and recall in user interfaces. *Nielsen Norman Group*, 2014.

[18] Stuart K Card, M Pavel, and JE Farrell. Window-based computer dialogues. In *Proc. Interact*, volume 4, pages 239–243, 1984.

[19] J. P. Chin, V. A. Diehl, and L. K. Norman. 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*, pages 213–218, New York, New York, USA, 1988. ACM Press.

[20] Tony Clark and Pierre-Alain Muller. Exploiting model driven technology: a tale of two startups. *Software & Systems Modeling*, 11(4):481–493, 2012.

[21] Andy Cockburn, Amy Karlson, and Benjamin B Bederson. A review of overview+ detail, zooming, and focus+context interfaces. *ACM Computing Surveys (CSUR)*, 41(1):2, 2009.

[22] Gilbert Cockton, Marta Lárusdóttir, Peggy Gregory, and Åsa Cajander. Integrating User-Centred Design in Agile Development. pages 1–46. Springer, 2016.

[23] Eleni Danili and Norman Reid. Cognitive factors that can potentially affect pupils' test performance. *Chemistry Education Research and Practice*, 7(2):64–83, 2006.

[24] David Dietrich and Joanne M. Atlee. A mode-based pattern for feature requirements, and a generic feature interface. In *21st IEEE International Requirements Engineering Conference, RE 2013 - Proceedings*, pages 82–91. IEEE, jul 2013.

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

[26] Sven Efftinge and Markus Völter. oAW xText: A framework for textual DSLs. In *Workshop on Modeling Symposium at Eclipse Summit*, volume 32, page 118, 2006.

[27] Alexander Egyed. Instant consistency checking for the uml. In *Proceedings of the 28th international conference on Software engineering*, pages 381–390. ACM, 2006.

[28] Colin Eles and Mark Lawford. A tabular expression toolbox for Matlab/Simulink. In *NASA Formal Methods Symposium*, volume 6617 of *Lecture Notes in Computer Science (LNCS)*, pages 494–499. 2011.

[29] Christian F. J. Lange. *Assessing and Improving the Quality of Modeling.* Ph.d., Technische Universiteit Eindhoven, Netherlands, 2007.

[30] Adrian Fernandez, Silvia Abrahão, Emilio Insfran, and Maristella Matera. Usability Inspection in Model-Driven Web Development: Empirical Validation in WebML. pages 740–756. Springer, Berlin, Heidelberg, 2013.

[31] Andy Field. *Discovering statistics using IBM SPSS statistics.* sage, 2013.

[32] Robert France, Jim Bieman, and Betty HC Cheng. Repository for model driven development (remodd). In *International Conference on Model Driven Engineering Languages and Systems*, pages 311–317. Springer, 2006.

[33] Miguel a. Garzon, Hamoud Aljamaan, and Timothy C. Lethbridge. Umple: A framework for Model Driven Development of Object-Oriented Systems. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pages 494–498, 2015.

[34] Sébastien Gérard, Cédric Dumoulin, Patrick Tessier, and Bran Selic. Papyrus: A UML2 tool for domain-specific language modeling. In *Proceedings of the International Dagstuhl Conference on Model-Based Engineering of Embedded Real-Time Systems (MBEERTS'07)*, pages 361–368. Springer-Verlag, 2007.

[35] Parisa Ghazi, Norbert Seyff, and Martin Glinz. FlexiView: A Magnet-Based Approach for Visualizing Requirements Artifacts. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, volume 9013 of *Lecture Notes in Computer Science (LNCS)*, pages 262–269. 2015.

[36] Martin Glinz, Stefan Berner, and Stefan Joos. Object-oriented modeling with adora. volume 27, pages 425–444. Elsevier, 2002.

[37] T.R.G. Green and M. Petre. Usability analysis of visual programming environments: A 'cognitive dimensions' framework. *Journal of Visual Languages Computing*, 7(2):131 – 174, 1996.

[38] Jonathan Grudin. *Why CSCW applications fail: Problems in the design and evaluation of organizational interfaces*. Microelectronics and Computer Technology Corporation, 1988.

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

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

[41] Markus Herrmannsdoerfer, Daniel Ratiu, and Guido Wachsmuth. Language evolution in practice: The history of GMF. In *International Conference on Software Language Engineering*, pages 3–22. Springer, 2009.

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

[43] James H Hill. 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)*, pages 120–129. IEEE, 2011.

[44] Zbigniew Huzar, Ludwik Kuzniarz, Gianna Reggio, and Jean Louis Sourrouille. Consistency Problems in UML-Based Software Development. In *UML Modeling Languages and Applications*, volume 3297, pages 1–12. Springer Berlin Heidelberg, 2004.

[45] No Magic Inc. Magicdraw, uml. 2013.

[46] Rodi Jolak, Truong Ho-Quang, Michel RV Chaudron, and Ramon RH Schiffelers. Model-based software engineering: A multiple-case study on challenges and development efforts. In *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, pages 213–223. ACM, 2018.

[47] Huzefa Kagdi and Jonathan I Maletic. Onion graphs for focus+ context views of uml class diagrams. In *4th International Workshop on Visualizing Software for Understanding and Analysis*, pages 80–87. IEEE, 2007.

[48] Stuart Kent. Model driven engineering. In *International Conference on Integrated Formal Methods*, pages 286–298. Springer, 2002.

[49] Mik Kersten. *Focusing knowledge work with task context.* PhD thesis, University of British Columbia, 2007.

[50] Alex Kirlik. Everyday life environments. *A companion to cognitive science*, pages 702–712, 2017.

[51] Andrew J Ko, Shriram Krishnamurthi, Gail Murphy, and Janet Siegmund. Human-Centric Development of Software Tools (Dagstuhl Seminar 15222). *Dagstuhl Reports*, 5(5):115–132, 2016.

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

[53] Samuel Lahtinen and Jari Peltonen. Adding speech recognition support to uml tools. *Journal of Visual Languages Computing*, 16(1):85 – 118, 2005. 2003 IEEE Symposium on Human Centric Computing Languages and Environments.

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

[55] Christian F J Lange and Michel R V Chaudron. Effects of Defects in UML Models: An Experimental Investigation. In *Proceedings of the 28th International Conference on Software Engineering*, pages 401–411, 2006.

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

[57] Hung Ledang and Jeanine Souquières. Contributions for modelling uml state-charts in b. In *International Conference on Integrated Formal Methods*, pages 109–127. Springer, 2002.

[58] A. Lédeczi, A. Bakay, M. Maróti, P. Völgyesi, G. Nordstrom, J. Sprinkle, and G. Karsai. Composing domain-specific design environments. *Computer*, 34(11):44–51, 2001.

[59] V. Levenshtein. Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission*, 1:8–17, 1965.

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

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

[62] James R Lewis. Measuring perceived usability: The csuq, sus, and umux. *International Journal of Human–Computer Interaction*, 34(12):1148–1156, 2018.

[63] Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

[64] Daniel Lucrédio, Renata P de M Fortes, and Jon Whittle. Moogle: A model search engine. In *International Conference on Model Driven Engineering Languages and Systems*, pages 296–310. Springer, 2008.

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

[66] I Scott MacKenzie. *Human-computer interaction: An empirical research perspective.* Newnes, 2012.

[67] David McNeish and Martin Maguire. A participatory approach to helicopter user interface design. 2019.

[68] Antkiewicz Michałand, Kacper Bak, Alexandr Murashkin, Rafael Olaechea, Jia Hui (Jimmy) Liang, Krzysztof Czarnecki, Michał Antkiewicz, Kacper Bąk, Alexandr Murashkin, Rafael Olaechea, Jia Hui (Jimmy) Liang, and Krzysztof Czarnecki. Clafer tools for product line engineering. In *Proceedings of the 17th International Software Product Line Conference co-located workshops on - SPLC '13 Workshops*, page 130, New York, New York, USA, 2013. ACM Press.

[69] Daniel Moody. The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on software engineering*, 35(6):756–779, 2009.

[70] Andreas Muelder. Yakindu Statechart Modeling Tools, 2011.

[71] 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. The relevance of model-driven engineering thirty years from now. pages 183–200, 2014.

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

[73] Bashar Nuseibeh, Steve Easterbrook, and Alessandra Russo. Making inconsistency respectable in software development. *Journal of Systems and Software*, 58(2):171–180, 2001.

[74] Object Management Group. OMG unified modeling language TM (OMG UML), superstructure v.2.3. *InformatikSpektrum*, (March):758.

[75] Visual Paradigm. Visual paradigm for uml. *Visual Paradigm for UML-UML tool for software application development*, page 72, 2013.

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

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

[78] Jakob Pietron, Alexander Raschke, Michael Stegmaier, Matthias Tichy, and Enrico Rukzio. A Study Design Template for Identifying Usability Issues in Graphical Modeling Tools. Technical report.

[79] Parsa Pourali. Tooling advances inspired to address observed challenges of developing uml-like models when using modelling tools. In *Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, pages 168–173. ACM, 2018.

[80] Parsa Pourali and Joanne M Atlee. An empirical investigation to understand the difficulties and challenges of software modellers when using modelling tools. In *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, pages 224–234. ACM, 2018.

[81] Parsa Pourali and Joanne M Atlee. A focus+context approach to alleviate cognitive challenges of editing and debugging uml models. In *Proceedings of the 22th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. IEEE, 2019.

[82] Parsa Pourali and Joanne M Atlee. UCAnDoModels: A context-based model editor for editing and debugging UML class and state-machine diagrams. In *Proceedings of the 22th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. IEEE, 2019.

[83] Parsa Pourali and Joanne M. Atlee. 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, 2018.

[84] Microsoft Press. *Microsoft Visual InterDev 6.0 Programmer's Guide*. Microsoft Press, 1998.

[85] Paul Ralph. Toward a theory of debiasing software development. In *EuroSymposium on Systems Analysis and Design*, pages 92–105. Springer, 2011.

[86] G Ramesh, TV Rajini Kanth, and A Ananda Rao. Extensible real time software design inconsistency checker: A model driven approach. In *Proceedings of the International MultiConference of Engineers and Computer Scientists (IMECS)*, volume 1, 2016.

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

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

[89] Pascal Roques. MBSE with the ARCADIA Method and the Capella Tool. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, Toulouse, France, January 2016.

[90] Dan Rubel, Jaime Wren, and Eric Clayberg. *The Eclipse Graphical Editing Framework (GEF)*. Addison-Wesley Professional, 2011.

[91] Elizabeth B-N Sanders and Pieter Jan Stappers. Co-creation and the new landscapes of design. *Co-design*, 4(1):5–18, 2008.

[92] Jeff Sauro and James R Lewis. *Quantifying the user experience: Practical statistics for user research*. Morgan Kaufmann, 2016.

[93] D.L. Schacter, D. T. Gilbert, and D. M. Wegner. *Psychology (2nd Edition)*. Worth, New York, 2011.

[94] Markus Scheidgen. Integrating Content Assist into Textual Modelling Editors. *Modellierung 2008, 12.-14. März 2008, Berlin*, 127:121–131, 2008.

[95] S. Sen, B. Baudry, and H. Vangheluwe. Towards Domain-specific Model Editors with Automatic Model Completion. *Simulation*, 86(2):109–126, feb 2010.

[96] Pourya Shaker. *A feature-oriented modelling language and a feature-interaction taxonomy for product-line requirements*. PhD thesis, University of Waterloo, 2013.

[97] Pourya Shaker, Joanne M. Atlee, and Shige Wang. A feature-oriented requirements modelling language. In *20th IEEE International Requirements Engineering Conference, RE 2012*, pages 151–160. IEEE, September 2012.

[98] Ben Shneiderman. Software psychology. *Winthrop, Cambridge, Mass*, 48:161–172, 1980.

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

[100] Neil R. Smalheiser. Chapter 12 - nonparametric tests. In Neil R. Smalheiser, editor, *Data Literacy*, pages 157 – 167. Academic Press, 2017.

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

[102] Monique Snoeck, Cindy Michiels, and Guido Dedene. Consistency by construction: the case of merode. In *International Conference on Conceptual Modeling*, pages 105–117. Springer, 2003.

[103] Fábio Soares, João Araújo, and Fernando Wanderley. VoiceToModel. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing - SAC '15*, pages 1350–1357, New York, New York, USA, 2015. ACM Press.

[104] OMG Available Specification. Omg unified modeling language (omg uml), superstructure, v2. 1.2. *Object Management Group*, 70, 2007.

[105] Friedrich Steimann and Bastian Ulke. Generic model assist. In Ana Moreira, Bernhard Schätz, Jeff Gray, Antonio Vallecillo, and Peter Clarke, editors, *International Conference on Model Driven Engineering Languages and Systems (MODELS)*, volume 8107 of *Lecture Notes in Computer Science (LNCS)*, pages 18–34. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[106] Alistair Sutcliffe. On the effective use and reuse of HCI knowledge. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(2):197–221, 2000.

[107] Rini van Solingen, Vic Basili, Gianluigi Caldiera, H. Dieter Rombach, Rini van Solingen (Revision), Vic Basili (Original article, 1994 ed.), Gianluigi Caldiera (Original article, 1994 ed.), and H. Dieter Rombach (Original article, 1994 ed.). Goal Question Metric (GQM) Approach. In *Encyclopedia of Software Engineering*. John Wiley & Sons, Inc., Hoboken, NJ, USA, jan 2002.

[108] Vladimir Viyović, Mirjam Maksimović, and Branko Perišić. Sirius: A rapid development of DSM graphical editor. In *INES 2014 - IEEE 18th International Conference on Intelligent Engineering Systems, Proceedings*, pages 233–238. IEEE, jul 2014.

[109] Earl L Wiener and Renwick E Curry. Flight-deck automation: Promises and problems. *Ergonomics*, 23(10):995–1011, 1980.

[110] Min Zeng, Pan Liu, and Huaikou Miao. The Design and Implementation of a Modeling Tool for Regular Expressions. In *3rd International Conference on Advanced Applied Informatics (IIAI)*, pages 726–731. IEEE, 2014.

Appendices

Appendix A

FORML Grammar Defined in Xtext

```
grammar uw.cs.watform.forml.Forml with org.eclipse.xtext.common.Terminals

generate forml "http://www.cs.uw/watform/forml/Forml"
import "http://www.eclipse.org/emf/2002/Ecore" as ecore

System:
    {System}
    ((worldmodel=WorldModel)? & (behaviourmodel=BehaviourModel)?);

/**
 * World Model
*/
WorldModel:
    {WorldModel} ('World' '{'
    (concepts+=Concept*
    constraints=Constraint?)
    '}');

Constraint:
    {Constraint} 'Constraints' '{' (predicates+=Predicate* &
    macro+=Macro*) '}';

ExprRef:
    Concept|Attribute
;

Concept returns ExprRef:
    Entity | Relationship | SPL | Message | Feature | Enumeration |
    UndefinedType;

Relationship returns Concept:
    Association | Aggregation | Composition;

Roleable returns Concept:
    Entity | Feature;

Entity returns Roleable:
    {Entity} 'abstract'? (ctrl?='ctrl')? 'entity' name=ID ('extends' superType=[Entity])? ('{'
attributes+=Attribute*
    functions+=Function* '}')?;

Association returns Relationship:
    {Association} (ctrl?='ctrl')? 'association' name=ID '{' (attributes+=Attribute* &
(srcRole=Role & desRole=Role)) '}';

Attribute returns ExprRef:
    {Attribute} (ctrl?='ctrl')? 'attribute' name=ID ('[' multiplicity=Multiplicity ']')? ':'
type=AttributeType?;

AttributeType:
    ReferenceType | IntType | BoolType |StringType| UndefinedType;

StringType:
    {StringType} "string" | "String";


Function returns ExprRef:
    {Function} 'function' name=ID ':' (type=AttributeType)?;

IntType:
    {IntType} "int" | "Integer";

BoolType:
    {BoolType} "bool" | "Boolean";

ReferenceType:
    ref=[Concept];

UndefinedType returns Concept:
    {UndefinedType} 'undefined_type' name=ID;

Multiplicity:
    (value=INT | many='*') | cardinality=Cardinality;

Cardinality:
    (lower=INT '..' (upper=INT | many='*'));


Role returns Role:
    'role' {Role} name=ID ('[' multiplicity=Multiplicity ']')? ':' type=[Roleable];

Aggregation returns Relationship:
    {Aggregation} (ctrl?='ctrl')? 'aggregation' name=ID '{' ('whole' whole=Decl & 'part'
part=Decl) '}';

Composition returns Relationship:
    {Composition} (ctrl?='ctrl')? 'composition' name=ID '{' ('whole' whole=Decl & 'part'
part=Decl) '}';

Decl:
    {Decl} name=ID ('[' multiplicity=Multiplicity ']')? ':' type=[Entity];

CompDecl returns Decl:
    {CompDecl} name=ID ':' type=[Entity];

IdList:
    ID (',' ID)*;


Message returns Concept:
    (Input | Output);

Input returns Message:
    {Input} 'input' name=ID '{' ('type' type=AttributeType)? attributes+=Attribute* ('to'
(tos+=[Feature])*)? '}';

Output returns Message:
    {Output} 'output' name=ID '{' ('type' type=AttributeType)? attributes+=Attribute* ('from'
(froms+=[Feature])*)? '}';

InputList:
    {InputList} 'inputs' '{' (inputs+=RefMessage (',' inputs+=RefMessage)*)? '}';

OutputList:
    {OutputList} 'outputs' '{' (outputs+=RefMessage (',' outputs+=RefMessage)*)? '}';

RefMessage:
    {RefMessage} refMsg=[Message];


Feature returns Roleable:
    {Feature} 'feature' name=ID ('{' (attributes+=Attribute* (inputlist=InputList)? (outputlist=OutputList)?) '}')?;

SPL returns Node:
    {SPL} 'SPL' name=ID '{' (featureNodes+=FeatureNode* xors+=XORNode* ands+=AndNode* ors+=OrNode*) '}';

XORNode returns Node:
    {XORNode} 'XOR' '{' featureNodes+=FeatureNode featureNodes+=(FeatureNode)+ '}';

AndNode returns Node:
    {AndNode} 'AND' '{' featureNodes+=FeatureNode featureNodes+=(FeatureNode)+ '}';

OrNode returns Node:
    {OrNode} 'OR' '{' featureNodes+=FeatureNode featureNodes+=(FeatureNode)+ '}';


FeatureNode returns Node:
    {FeatureNode} 'FeatureNode' feature=[Feature] ('[' multiplicity=Multiplicity ']')? ('{' (featureNodes+=FeatureNode*
    xors+=XORNode* ands+=AndNode* ors+=OrNode*) '}')?;

Enumeration returns Concept:
    {Enumeration} 'enum' name=ID '{' (constant+=EnumConst (',' constant+=EnumConst)*) '}';

EnumConst returns ExprRef:
    {EnumConst} name=ID;


BehaviourModel:
    {BehaviourModel} 'Behaviour' '{' featuremodules+=FeatureModule* '}';

FeatureModule:
    'FeatureModule' featureref=[Roleable] '{' (constraints=Constraint? & statemachine+=StateMachine* &
    fragments+=Fragment*) '}';

StateMachine:
    'statemachine' name=ID '{' (regions+=Region+ & transitions+=Transition*) '}';

InitState:
    {InitState} 'init' '=' stateref=[State]?;

State returns Component:
    {State} 'state' name=ID '{' regions+=Region* '}';

Region returns Component:
    {Region} 'region' name=ID '{' init=InitState? states+=State* '}';

Transition:
    {Transition} 'transition' name=ID priority=Priority? ':' src=[State|QualifiedName] '->' dst=[State|QualifiedName] '{'
    trig=Trigger? guard=Guard? (list=WCAList)? '}';


Priority:
    '>' translist=TransList;

TransList:
    transitions+=[Transition] (',' transitions+=[Transition])*;

Guard:
    {Guard} ('[' predicate=Predicate? ']' | '[]');


Trigger:
    {Trigger} 'event:' (override=Override | wce=WCE)?
;

Override:
    'override' '(' transition=[Transition] ')';


WCE:
    (addobj=[ExprRef|QualifiedName] '+') | (attr=[Attribute|QualifiedName]) | (remobj=[ExprRef|QualifiedName] '-');

WCAList:
    {WCAList} '/' (actions+=WCA (',' actions+=WCA)*)?;

WCA:
    {WCA} name=ID '{' action=Action?

    '}';

Action:
    {Action} '+' type=[ExprRef|QualifiedName]
    // Bang is for outputting message?
    | '!' out=[Output] '(' list+=AssignList ')' | '-' minexpr=setExpr | (lvalue=setExpr)? ':=' (rvalue=setExpr)? | (lvalue=intExpr)
    ':=' (rvalue=intExpr);

AssignList:
    list+=Assign (',' list+=Assign)*;

Assign:
    ref=[ExprRef] '=' sexpr=(setExpr | intExpr);

/**
 * Expressions
 */
Macro:
    'let' name=ID '=' expr=(setExpr | intExpr);

Predicate:
    (logicop=('no' | 'lone' | 'one' | 'some' | 'all') var=Variable ':' sexpr=setExpr '|')? quantPred=notPred;
```

```

notPred:
    'not' pred=Predicate | pred=andPred;

andPred:
    preds+=impPred ('and' preds+=impPred)*;

impPred:
    preds+=orPred (logicop+=('implies' | 'iff') preds+=orPred)*;

orPred:
    preds+=basePred ('or' preds+=basePred)*;

basePred:
    logicop=('no' | 'lone' | 'one' | 'some' | 'all') cardpred=setExpr | setlhs=predExpr
logicop=('in' | '=')
    setrhs=predExpr? | setlhs=predExpr logicop=('in' | '=') boolrhs=boolExpr | intlhs=intExpr
op=('=' | '<>' | '>' | '<'
    | '>=' | '=>' | '<=' | '=<') intrhs=intExpr? | '(' parenPred=Predicate ')';

boolExpr:
    isTrue='true' | isFalse='false';

predExpr:
    expr+=predExpr2 ('&' expr+=predExpr2)*;

predExpr2:
    expr+=predExpr3 (op+=('-' | '+') expr+=predExpr3)*;

predExpr3:
    base=setExprBase;

setExpr:
    expr+=setExpr2 ('&' expr+=setExpr2)*;

setExpr2:
    expr+=setExpr3 (op+=('-' | '+') expr+=setExpr3)*;

setExpr3:
    '(' paren=setExprBase ')' | base=setExprBase;

setExprBase:
    (atom=atomic) | unspec=unspecified;

intExpr:
    lhs=multExpr (op+=('+' | '-') rhs+=multExpr)*;

multExpr:
    lhs=intBase (op+=('*' | '/') rhs+=intBase)*;

intBase:
    '#'(atom=atomic)| num=INT;

atomic:
    none='none' |
    ref=[ExprRef|QualifiedName] '@pre'? | refs=[ExprRef|QualifiedName] 's' '@pre'?;

```

```
unspecified:
    ref=[ExprRef] '()' '@pre'?;

setOper:
    '+' | '-' | '&';

Variable returns ExprRef:
    {Variable} name=ID;

/**
 * FRAGMENTS
 */
Fragment:
    {Fragment} 'fragment' name=ID '{' fragmentType+=FragmentType* '}';

FragmentType:
    (state=StateContext | region=RegionContext | frag=TranFragment |
statemachine=StateMachineContext) &
    transitions+=Transition*;

StateContext:
    {StateContext} 'StateFragment' '{' 'state' ref=[State|QualifiedName] regions+=Region* '}';

RegionContext:
    {RegionContext} 'RegionFragment' '{' 'region' ref=[Region|QualifiedName] states+=State*
'}';

StateMachineContext:
    {StateMachineContext} 'SMFragment' '{' 'statemachine' ref=[StateMachine|QualifiedName]
state+=State*;

TranFragment:
    'TransitionFragment' '{' ref=[Transition|QualifiedName] ':' trig=Trigger? fragType=(Guard)
    ('/' list=WCAList? fragList=WCAFragmentList?)? '}';

WCAFragmentList:
    frag+=WCAFragment (',' frag+=WCAFragment)*;

WCAFragment:
    {WCAFragment} ref=[WCA|QualifiedName] ':' fragType=(Guard);

QualifiedName:
    ID ('.' ID)*;
```

Appendix B

Assignment Used for the Pre-Study Phase

CS 445 / ECE 451 / CS 645 / SE 463
Software Requirements Specification and Analysis

Assignment 3: State-Machine Modelling, Temporal Logic

Due Date: Tuesday, March 11 at 12:00pm

(Late Date: Thursday, March 13 at 12:00pm)

Submission

Combine your models and answers in a single PDF file and email it to cs445@student.cs.uwaterloo.ca.

1) State Machine Model

You are to create a UML State-Machine model of the proposed system to enforce the parking policies at the Bauer complex. Bauer provides parking for two types of users: customers and employees. There are two levels of parking: surface-level parking and underground parking. Surface-level parking is for customers only. They pay a $2 flat fee for parking. Employees park for free, but only if they park underground. Underground parking is also open to customers in the evenings and on weekends, but of course customers must still pay ($2) to use it.

Gates control access to both levels of parking. Each gate controls traffic flowing in a single direction (e.g., traffic entering the complex), and it opens only for cars that approach the gate from the appropriate direction. Employees' cars have transponders that the gates can sense. Some gates open only for cars that have transponders. In addition, the system uses transponder data (IDs) to keep track of where employees' cars park. There are four gates that are controlled independently:

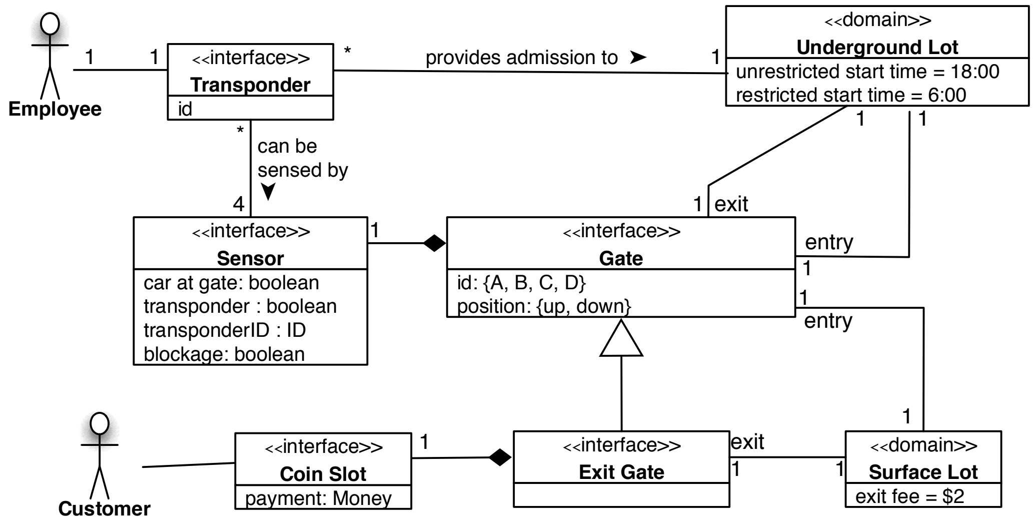
- Gate **A** controls entry into the Bauer complex. It opens whenever it senses a car approaching from outside the Bauer complex. Cars that pass through gate A enter the surface-level parking lot.
- Gate **B** controls access from surface-level parking to underground parking. Normally, it opens only if an approaching car has a transponder. But during evenings (18:00-6:00) and weekends (Friday 18:00 - Monday 6:00), it remains open to all cars.
- Gate **C** opens whenever it senses a car approaching from the underground parking lot.
- Gate **D** controls exits from the Bauer complex and collects parking fees. The gate opens whenever a $2 fee is paid (no change is provided). In addition, if an approaching car has a transponder, the gate opens if the car had parked underground (i.e., if the transponder triggered gate B since the car entered the complex).

Each of the gates closes 5 seconds after it opens — unless a blockage is detected, in which case it remains open and tries to close again after another 5 seconds. (Don't worry about modelling the delay of the gate opening and closing.)

On the next page is a domain model for the system based on the interface choices already made. The events, conditions, and actions in your state-machine model should be expressed in terms of elements in the domain model. You can declare events and abbreviations based on these elements, to simplify the expressions in your model.

For full credit, your model should make the most effective use of UML state-machine modelling constructs. You must use a software modelling tool or drawing tool to create the state machine model that you include in your submission.

1) State Machine Model (cont.)
Below is a domain model for the system that enforces the Bauer parking policies.



Appendix C

Materials for the Formative User Study

Recruitment Letter

Hello,

My name is Parsa Pourali, a PhD student in the department of Electrical and Computer Engineering working under the supervision of Prof. Joanne M. Atlee, a professor in the Cheriton School of Computer Science at the University of Waterloo. I am writing to ask for your participation in conducting a study that aims at understanding how to reduce the effort of creating and editing software models, through novel enhancements to software modelling tools.

Participation in this study involves coming into the laboratory and performing a few software modelling tasks. The tasks include developing class diagrams and state machines, such as setting the triggering events, guards and actions for transitions. Your activities will be audio-recorded as well as video-captured from the computer screen. We will measure the time that it takes you to fulfill each task and gauge your success or failure on a task. However, this study is meant to help us gauge the effectiveness of current modelling tools; it is not intended to test your individual performance in any way. Participation in this study would take approximately 60 to 90 minutes of your time. In appreciation of your time commitment, you will receive an honorarium of $20 and will be given the chance to enter into a draw and win a prize of $200 gift card. Odds for the draw are equal for all the participants which is approximately 1 in 30. I would like to assure you that the study has been reviewed and received ethics clearance through the University of Waterloo Research Ethics Committee.

If you are interested in participating, please perform the recruitment screening procedure by filling the screening form at: **[SURVEY MONKEY LINK]**. We will then take your information to see whether you fit the study, and will contact you by email to schedule your laboratory session. You can cancel your appointment by sending me an email at ppourali@uwaterloo.ca. Please note that, the honorarium amount of $20 will be paid to you only if you pass the screening questions and attend to the study. All screening data will be stored securely, if you do not fit the study or decide not to participate your screening data will be deleted. We will also notify you by email.

Please note that, you will be completing the screening by an online survey operated by SurveyMonkey. When information is transmitted over the internet privacy cannot be guaranteed. There is always a risk your responses may be intercepted by a third party (e.g., government agencies, hackers). SurveyMonkey temporarily collects your computer IP address to avoid duplicate responses in the dataset but will not collect information that could identify you personally.

If you have any questions regarding this study, or would like additional information to assist you in reaching a decision about participation, please contact Parsa Pourali by email at ppourali@uwaterloo.ca.

Sincerely,

Parsa Pourali

A Research Study on Understanding and Learning About the Efforts and Challenges of using UML Modelling tools

Recruitment Screener

Hello,

This is a recruitment screener for a study on observing modellers and understanding their difficulties and challenges to improve the quality of modelling tools. It is important to note that you will not be evaluated in any way and it is the tool that is under scrutiny, not you. The activity takes place in the University of Waterloo, room DC2551B. Participation is on a paid basis.

Are you interested in participating?

- **Yes.**
- **No.** *(Will thank the person and end the recruitment.)*

We have a few questions to ask you to see whether you fit the profile of the individuals we need for this study. After you answer the questions, we will take your information for review, and then we will contact you by email to let you know whether your background is a good fit, and to schedule your laboratory session.

A. Background and Demographic Information
Name: *Your name will not be associated with your data.*
Email:

1. Are you currently enrolled as a student at the University of Waterloo?
—Graduate —Undergraduate —No

2. Department/Major? ——————————————

3. Have you taken a software engineering course that includes UML modelling?
—Yes —No

4. How would you rate your familiarity with UML?
(0)Unfamiliar
(1)Fairly Familiar (Novice)
(2)Familiar
(3)Very Familiar
(4)Strongly Familiar (Experienced)

5. How would you rate your familiarity with UML Class Diagrams?
(0)Unfamiliar
(1)Fairly Familiar (Novice)
(2)Familiar
(3)Very Familiar
(4)Strongly Familiar (Experienced)

6. How would you rate your familiarity with UML State Diagrams?
(0)Unfamiliar

(1)Fairly Familiar (Novice)
(2)Familiar
(3)Very Familiar
(4)Strongly Familiar (Experienced)

7. Do you have any experience with using modelling tools?
—Yes —No ;

8. (If Yes on Q7, survey monkey will show this question) In total, for how long have you had the experience i.e. how old is your experience (in month) ——

9. (If Yes on Q7, survey monkey will show this question) How frequently did you use the tools?
– Several times a day
– 3-5 days a week
– 1-2 days a week
– Every few weeks
– Less often

10. (If Yes on Q7, survey monkey will show this question) Please list the name of the tools with which you have experience: ——————

11. (If Yes on Q7, survey monkey will show this question) How much of your experience with the modelling tools was in an industrial setting (in month)? ——

12. In this study, you will be asked to use one of the modelling tools of your choice. What modelling tool do you want us to prepare for you to use (e.g., MagicDraw, VisualParadigm, Papyrus, ArgoUML, etc.)? —————-

B. Confidentiality agreement, and permission to record audio and screen-capture
We also have some questions to make sure you understand and are comfortable with our procedure before you come in:

1. Are you willing to sign a standard consent form, which acknowledges that you agree to participate?
—Yes —No

2. Are you willing to be audio-recorded and your activities with the tool be screen-captured? (The purpose is so we can go back and capture more detailed notes. Videos are seen internally only by members of the research team - mainly the research student - who are interested in what you have to say.)
—Yes —No

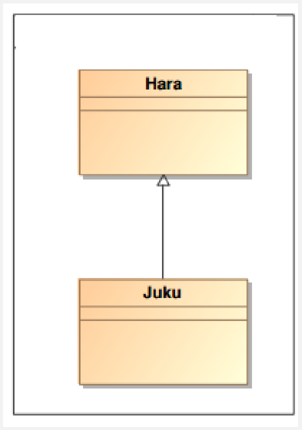
3. The compensation is 20.00 CAD and the chance to enter into a draw to win a prize of $200 gift card. Are you willing to participate based on the compensation?
—Yes —No

C. Screening Exercise: UML Knowledge Assessment

We would like to ask you a few questions[a] to assess if you are a good match with our study.

Question 1
What type of relationship is needed to represent the relationship between students and the courses they are enrolled in a university?
1) A one-to-one association.
2) A one-to-one composition.
3) A one-to-many association.
4) A one-to-many composition.
5) A many-to-many association.
6) A many-to-many composition

Question 2
Exhibit:



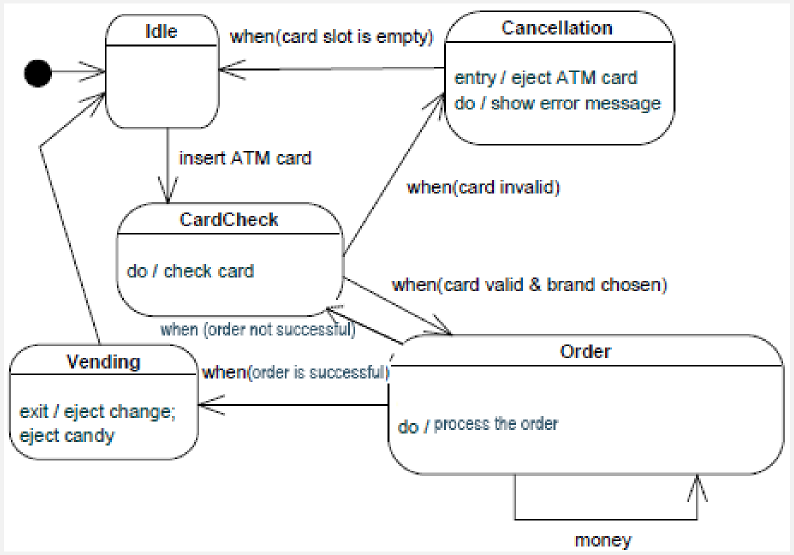
Which of the following is true?
1) Juku is a subclass of Hara.
2) This is NOT a valid UML class diagram.
3) Every Juku has a reference to at least one Hara.
4) Juku is a subclass of Hara and at least one other class

Question 3
You can model the following situations with a state machine diagram:
1) Activities that are executed while the object is in a certain state or while a transition is occurring.
2) Events that trigger transitions.
3) The states that the object of a class can have.
4) Possible transitions from one state to another.
5) All of the above items.

Question 4
Which of the following statements are true?

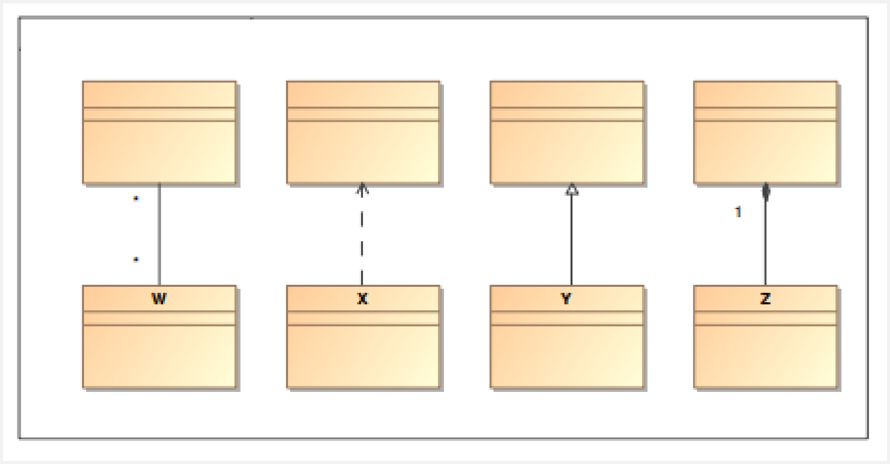
1) Every class in a class diagram may or may not have attributes and operations.
2) An attribute can have a type.
3) Operations may have parameters and return values.
4) All of the above items.

Question 5
Exhibit:



You are given the above state machine diagram. Which of the following statements about allowable state order is true?
1) money is an event.
2) money is a state.
3) When money occurs, the system will go to its Vending state.
4) None of the above.

Question 6
Exhibit:



Which class has a superclass relationship?
1) W
2) X
3) Y
4) Z

Question 7
What does the syntax for labelling a transition look like?
1) event [guard] / action
2) [guard] action / event
3) action [guard] / event
4) [action] guard / event
5) [action] event / guard

Question 8
You want to model the following situation: A home delivery service has the two states Wait and Deliver. At the beginning, state Wait is active. As soon as a customer has ordered a product, a transition to state Deliver takes place. During the transition from state Wait to state Deliver, the order is processed. State Deliver stays active until the product has been delivered to the customer, triggering a transition to state Wait. Which transition expression would you select from the list below for the transition from state Wait to state Deliver?
1) order received / process order.
2) / process order
3) order received [process order]
4) [order received] / order is processed
5) [order] / process order
6) [order received] / process order

[a]The questions are originally taken from UML knowledge assessment exercise from the online sample practice tests for the Sun Certified Java Associate exams and the UML quiz website: http://elearning.uml.ac.at.

Statement of Informed Consent

By signing this consent form, you are not waiving your legal rights or releasing the investigator(s) or involved institution(s) from their legal and professional responsibilities.

Title of the study: A research study on Understanding and Learning About the Efforts and Challenges of Using UML Modelling Tools.

Purpose: You have been asked to participate in a research study for understanding UML modellers and identifying the difficulties that they face when using UML modelling tools. By participating in this activity, you will help us enhance modelling tools and improve their usability, usefulness, and quality. This activity is meant to help us gauge the effectiveness of current modelling tools; it is not intended to test your individual performance in any way.

Procedure: You will be asked to perform a few modelling tasks related to developing UML class and state diagrams such as creating a transition expression in a state diagram. Also, you will be asked to think-aloud your thoughts when you are performing the given tasks. While you work, I will video-capture the computer screen, record your voice, and record some comments such as your time on tasks and task success or failure. It is expected that the study will take 60 to 90 minutes of your time. In appreciation of your time commitment, you will receive an honorarium of $20 and will be given the chance to enter into a draw and win a prize of $200 gift card. Odds for the draw are equal for all the participants which is approximately 1 in 30. The amount received is taxable. It is your responsibility to report this amount for income tax purposes.

Confidentiality: We will use the data you give us, along with the information we collect from other participants, to determine the difficulties that the modellers face while performing modelling tasks. To ensure confidentiality, we will not associate your name with your data. This session will be audio-record.

Benefits: Although the research may provide no direct personal benefit to you (as a participant), it will provide benefits to the academic community/society in a way that it will improve the quality of UML modelling tools and therefore alleviate the task of software modelling.

Risks: Generally, there are no known or anticipated risks associated with participation in this study. However, we think that because of the nature of the think-aloud protocol, you might feel that you are under the scrutiny and may feel nervous about your performance. We would like to re-emphasize that you will not be evaluated in any ways. It is the tool that is under scrutiny, not you.

Breaks: There will not be a scheduled break. However, you may take a break at any time.

Your Rights as a Participant: Your participation in this study is voluntary. You also have the right of freedom to withdraw. You may withdraw from the activity at any time without penalty. You will still receive the $20 if you withdraw during the session. However, you will not be eligible to enter into the draw. Moreover, you can request your data to be removed from the study up until 2018/02/28 as it is not possible to withdraw your data once papers and publications have been submitted to publishers.

I have read the information presented in the information letter about a study being conducted by Parsa Pourali, a PhD student of the Department of Electrical and Computer Engineering at the University of Waterloo. I have had the opportunity to ask any questions related to this study, and to receive satisfactory answers to my questions and any additional details I wanted. I am aware that I may withdraw from the study without penalty at any time by advising the researchers of this decision. I understand my work during the study will be observed and Parsa Pourali will be measuring the time that it takes me to fulfil my tasks.

☐ I agree to my interview being audio recorded to ensure accurate transcription and analysis.
☐ I agree to my working computer screen being video captured for the purpose of analysing my performance with the tools.

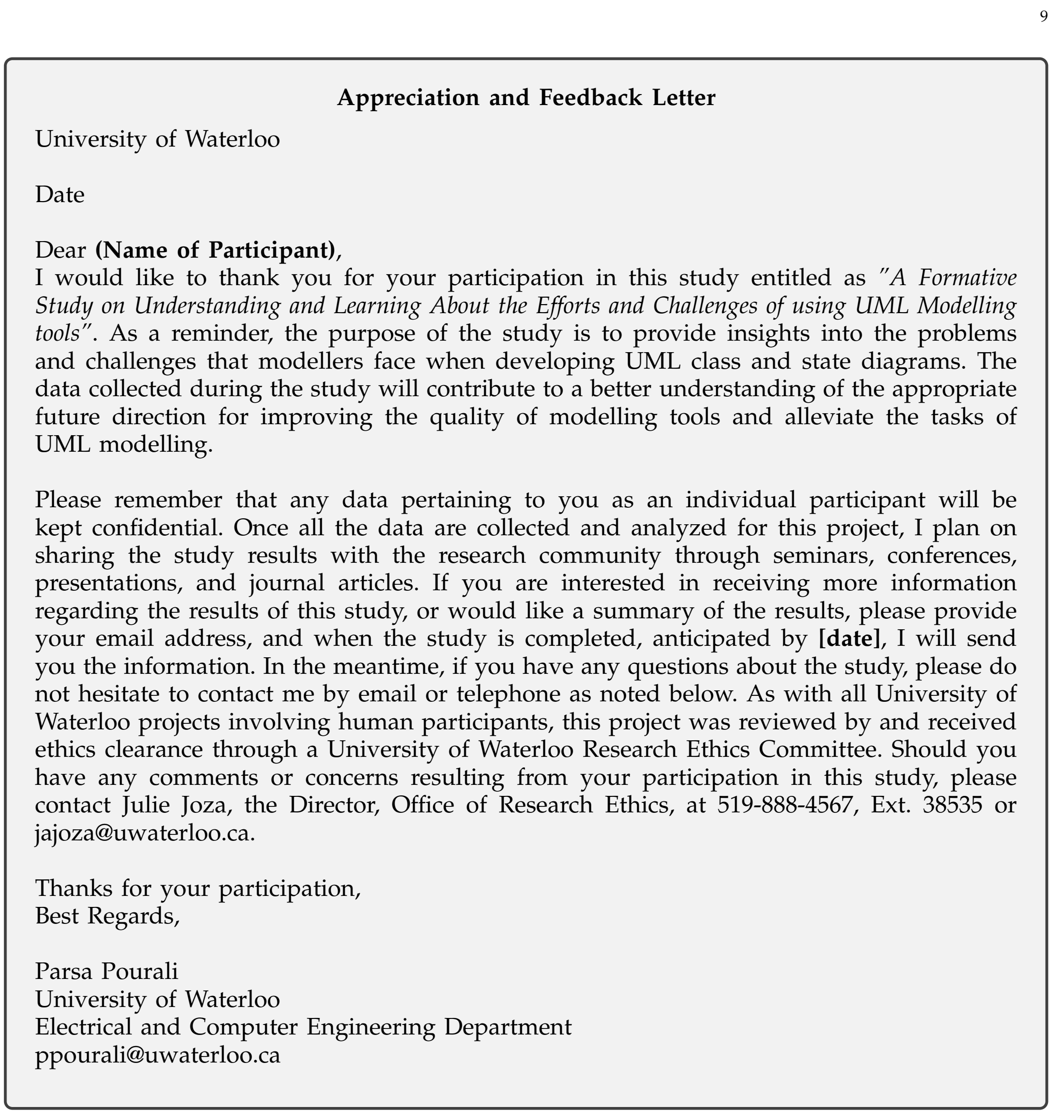
This study is funded under the grant of ORF-RE (Model-Based Software Engineering), and has been reviewed and received ethics clearance through a University of Waterloo Research Ethics Committee (ORE#22498). If you have questions for the Committee contact the Chief Ethics Officer, Office of Research Ethics, at 1-519-888-4567 ext. 36005 or oreceo@uwaterloo.ca.

If you have any questions regarding this study, or would like additional information to assist you in reaching a decision about participation, please contact Parsa Pourali by email at ppourali@uwaterloo.ca.

With full knowledge of all the foregoing, I agree, of my own free will, to participate in this study.

Print Name: – – – – – – – – – – – – – – – – – –

Signature of Participant: – – – – – – – – – – – – – –   Date: – – – – – – – –



Appreciation and Feedback Letter

University of Waterloo

Date

Dear **(Name of Participant)**,
I would like to thank you for your participation in this study entitled as *"A Formative Study on Understanding and Learning About the Efforts and Challenges of using UML Modelling tools"*. As a reminder, the purpose of the study is to provide insights into the problems and challenges that modellers face when developing UML class and state diagrams. The data collected during the study will contribute to a better understanding of the appropriate future direction for improving the quality of modelling tools and alleviate the tasks of UML modelling.

Please remember that any data pertaining to you as an individual participant will be kept confidential. Once all the data are collected and analyzed for this project, I plan on sharing the study results with the research community through seminars, conferences, presentations, and journal articles. If you are interested in receiving more information regarding the results of this study, or would like a summary of the results, please provide your email address, and when the study is completed, anticipated by **[date]**, I will send you the information. In the meantime, if you have any questions about the study, please do not hesitate to contact me by email or telephone as noted below. As with all University of Waterloo projects involving human participants, this project was reviewed by and received ethics clearance through a University of Waterloo Research Ethics Committee. Should you have any comments or concerns resulting from your participation in this study, please contact Julie Joza, the Director, Office of Research Ethics, at 519-888-4567, Ext. 38535 or jajoza@uwaterloo.ca.

Thanks for your participation,
Best Regards,

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

Study Execution Protocol

Before participant arrives:

- Keep the pen, payment forms and consent form ready (make sure there are extra ones as well).
- Put a note on the door showing that the study will be held in this office, so that when the participant arrives, he/she knows which room to enter or which door to knock on.
- Open and prepare the introduction video on the PC.
- Open and prepare the tools that the participant asked for, and set up the experiment models.
- Open the mouse highlighting tool and make sure it works properly.
- Assign a participant ID to the participant.
- Prepare the screen-capturing and voice-recording applications (also test them to make sure they work properly).

After participant arrives:

1) Explain who we are and the purpose of the study.

   Facilitator says: "Hello, my name is Parsa. I am a PhD student who is doing research on Model-Easing and Reducing the Effort of Software Modelling for Modellers. As part of my research work, I need to understand users (who are software modellers) to reduce the effort of creating and editing software models, through novel enhancements to software modelling tools. Today, you are here to help me identify some of the difficulties that a modeller may face when working with modelling tools."

2) Give and explain the forms (e.g., consent form).

   Facilitator says: "Before we go further, I would like to ask you to kindly read these forms which are letting you know about your rights as a participant. We will continue when you sign the forms. Thanks."

3) Open and show the modelling tool to the participant.

   Facilitator says: "Great! This is the modelling tool that you requested to work with. It is prepared and set up for the study and you will be working with it today."

4) Explain the introduction video.

   Facilitator says: "Now, I will play a video for you which explains the study in more detail. If necessary, please feel free to pause the video and ask your questions about the video."

5) Once the video has ended:

   Facilitator says: "Okay, you can now start your tasks. The tasks descriptions is prepared for you in a pdf file that you can find on the taskbar as well as the modelling tool that is also minimized and is prepared for you to work with."

Finishing Up:

1) As the participant finishes, thank them for their time, give them their payment and ask them to fill the letter of Acknowledgement of Receipt of Remuneration, and escort them out of the office.
2) Save all the works that have been done by the participant and put the forms in a safe place.
3) Backup the databases.
4) Save the video and audio files with the participant ID.

Preparation Script

Hello,

Thank you for accepting our invitation and for your participation in the study. The goal of the study is to observe and analyse modellers when they are performing their modelling tasks, and try to understand the most pressing difficulties and challenges that they might face when using modelling tools.

During the study, you will be given a set of modelling tasks to perform. You will perform the tasks given to you in the modelling tool that is prepared and is open for you to work. In the meantime, we will be capturing the computer screen for further analysis. Also, as you work on the tasks, I will ask you to do what we call 'think out loud'. What that means is that I want you to say out loud what you are thinking as you work. You might speak about the steps in the task as you complete them, as well as your expectations and evaluation statements. Let us show you what we mean by showing you a demo:

(playing a video about think aloud protocol)

Do you see what I mean about think out loud? I would like to remind you to please speak loudly since we will be recording your voice. Your voice remains confidential, and the researcher is the only person who will listen to it during later analysis.

The main task of the observer is to jot down what happens. In addition, we might prompt you by asking a few questions to make the session more effective. Such questions may include:

What are you thinking now?
Why did you do that?
please keep talking ...

Now, before we start the experiment, I would like to emphasize a few important notes to you:

1- In the course of the study, you are the expert.
2- Remember that you will not be evaluated in any way. It is the tool that is under scrutiny, not you.
3- You should at all times comment liberally on your actions, intentions and thoughts.
4- You should be at ease and relaxed. Please keep in mind that, the think-aloud method is informal, and the most effective single way to maximise its effectiveness is to create an informal atmosphere. That's why we want you to be relaxed. Again, it is the tool that is under scrutiny, not you.
5- You should try to find your own way as much as possible. You may ask questions at any point in the process, but I may not answer them or only answer those questions that are related to clarifying and understanding the application domain. I will only be able to give you the bare minimum of help. We apologise in advance for this.
6- You can stop a task at any time if it becomes uncomfortable.
7- I will not tell you when you have completed task; you must determine this on your own.
8- If a task is taking longer than what is expected, I might ask you to ignore the current task and move on to the next task.

Now, if you have any remaining questions, please ask them, otherwise, let's begin the tasks.
Thank you.

Task Description

A modeller is responsible for editing several types of models such as Class Diagram, Sequence Diagram, State Diagram, Activity Diagram, and Use-Case Diagram. Assume that as a modeller, you are given a description of a system to enforce the parking policies at the Bauer complex, and your task is to update the State Diagram. Although your task is only to update/complete the State Diagram, you may find it necessary to also update the existing Class Diagram (i.e. adding a few model elements). You need to set the different parts of the transitions, namely, Triggering Event, Guard or Action, based on the description that is given to you for each part. Please remember to say your thoughts (**especially your challenges and difficulties, as well as your expectations from the tool**) loudly so that we can record them clearly.

System Description: You are to create a UML State-Machine model of the proposed system to enforce the parking policies at the Bauer complex. Bauer provides parking for two types of users: customers and employees. There are two levels of parking: surface-level parking and underground parking. Surface-level parking is for customers only. They pay a $2 flat fee for parking. Employees park for free, but only if they park underground. Underground parking is also open to customers in the evenings, but of course customers must still pay ($2) to use it.

Gates control access to both levels of parking. Each gate controls traffic flowing in a single direction (e.g., traffic entering the complex), and it opens only for cars that approach the gate from the appropriate direction. Employees' cars have transponders that the gates can sense. Some gates open only for cars that have transponders. In addition, the system uses transponder data (IDs) to keep track of where employees' cars park. There are four gates that are controlled independently:

1) Gate **A** controls entry into the Bauer complex. It opens whenever it senses a car approaching from outside the Bauer complex. Cars that pass through gate A enter the surface-level parking lot.
2) Gate **B** controls access from surface-level parking to underground parking. Normally, it opens only if an approaching car has a transponder. But during evenings (18:00-6:00), it remains open to all cars.
3) Gate **C** opens whenever it senses a car approaching from the underground parking lot.
4) Gate **D** controls exits from the Bauer complex and collects parking fees. The gate opens whenever a $2 fee is paid (no change is provided). In addition, if an approaching car has a transponder, the gate opens if the car had parked underground (i.e., if the transponder triggered gate B since the car entered the complex).

Each of the gates closes 5 seconds after it opens, unless a blockage is detected, in which case it remains open and tries to close again after another 5 seconds. (Don't worry about modelling the delay of the gate opening and closing.)

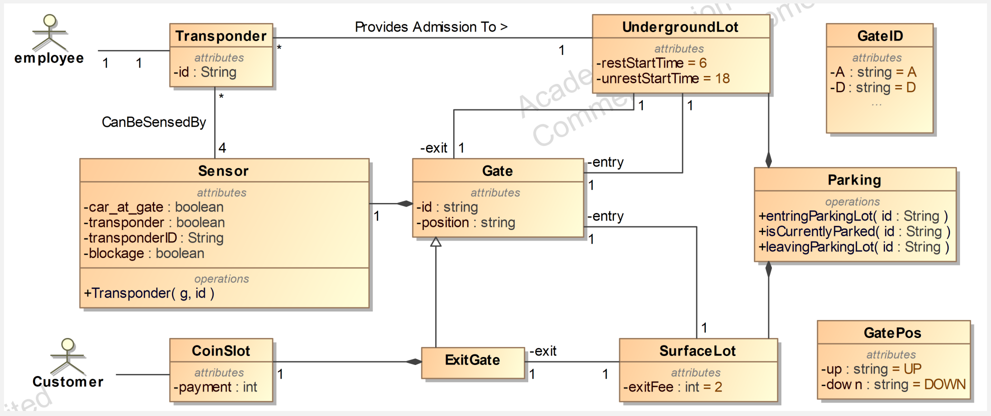


Fig. 1: Class Diagram for the Parking Lot application

Fig. 1 is the partial domain model for the system based on the interface choices already made. The events, conditions, and actions in your state-machine model should be expressed in terms of elements in the domain model. Moreover, the description of the domain model is given below. It helps you understand the domain, classes, attributes and the messages (operations) that you may need to know during the study. It is suggested that you take a brief look at the description before starting the tasks.

UndergroundLot: Represents the Underground parking of the parking lot.

- *restStartTime:* The Underground Lot has a restricted time for customers starting at 6:00am.
- *unrestStartTime:* The Underground Lot becomes unrestricted for customers starting at 18:00pm.

SurfaceLot: Represents the Surface-Level parking of the parking lot.

- *exitFee:* Represents the $2 parking fee that each customer must pay when exiting the lot.

GateID: Each Gate has an ID of either A, B, C, or D. GateID is an enum variable (class) that represents the id of each gate. Please note that, since the class diagram is not fully complete (intentionally), the id of B is not defined in this class.

GatePos: Gates can be either open or close, that is the gates' position can be either up or down. This enum class represents the allowable values for the gates' position.

Gate: Represents the Gates for both underground and surface-level parking lots.

- *id:* Each Gate has an id of the type GateID, that is a Gate can have an id of A, B, C, or D.
- *position:* Each Gate has a position attribute that is from the type of the class GatePos, and can have a value of up or down.

ExitGate: It shows the exit gate of the surface-level parking where the customer needs to put money in the coin slot.

CoinSlot: When exiting from the exit gate of the surface-level parking, the customer needs to pay the exit fee through a coin slot. The CoinSlot class represents the coin slot entity of the domain.

- *payment:* The amount that the customer has already inserted to the coin slot.

Sensor: Each Gate has a sensor the senses if a car is approaching the gate.

- *car_at_gate:* This is a boolean attribute that returns true if there is a car at the gate, and returns false if there is no car at the gate.
- *transponder:* Returns true if an approaching care has a transponder. Otherwise, returns false.
- *transponderID:* If an approaching care has a transponder, this attribute returns the ID of the transponder.
- *blockage:* Returns true in a case that a car has come through the gate, so that the gate does not close.
- *Transponder (g, id):* Returns true if a transponder (with id=id) is approaching gate **g** from the sensed direction.

Transponder: Each employee has a Transponder installed on their car. that provides admission to the underground parking and can be sensed by the sensors on the gates.

- *id:* Each Transponder has an ID.

Parking: This is a singleton class which records the states of cars with the transponder. Moreover, if a car enters to the parking, its transponder id should be recorded and be added to the collection of the parked cars' transponder ids. Also, when it leaves the parking, the car's transponder id must be removed from the list.

- *enteringParkingLot (id):* When a car is entering to the parking lot, this operation adds the transponder id of the car to the list of the transponder ids of the existing cars in the parking.
- *isCurrentlyParked (id):* This is a boolean operation which checks if a car with a transponder id had parked in the parking.
- *leavingParkingLot (id):* When a car is leaving the parking, the system uses this operation to remove the transponder id of the leaving car from the list of the transponder ids of the parked cars.

☐ I have read the system description carefully.

Fig. 2 shows a partial state machine for the system. As can be seen in the state machine, the diagram is complete for the gates A and C, whereas the gates B and D still need to be developed more completely.

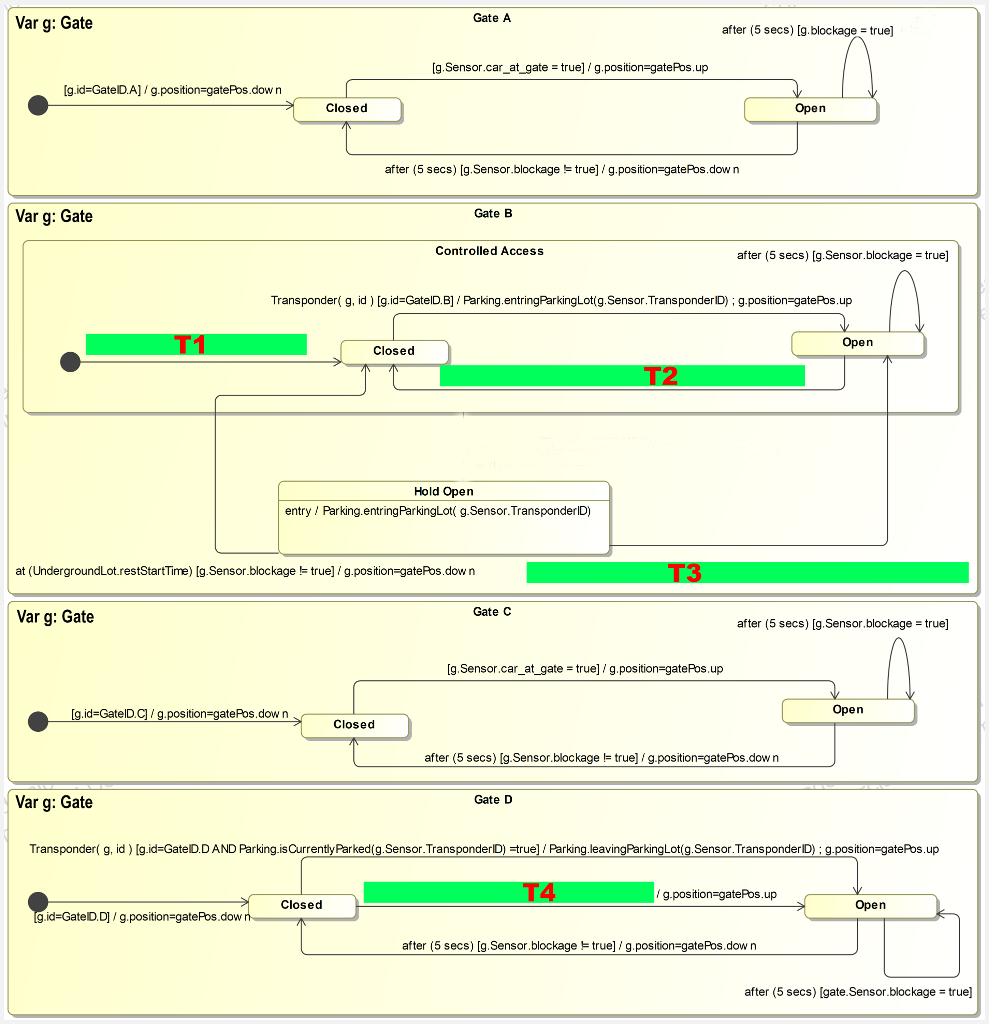
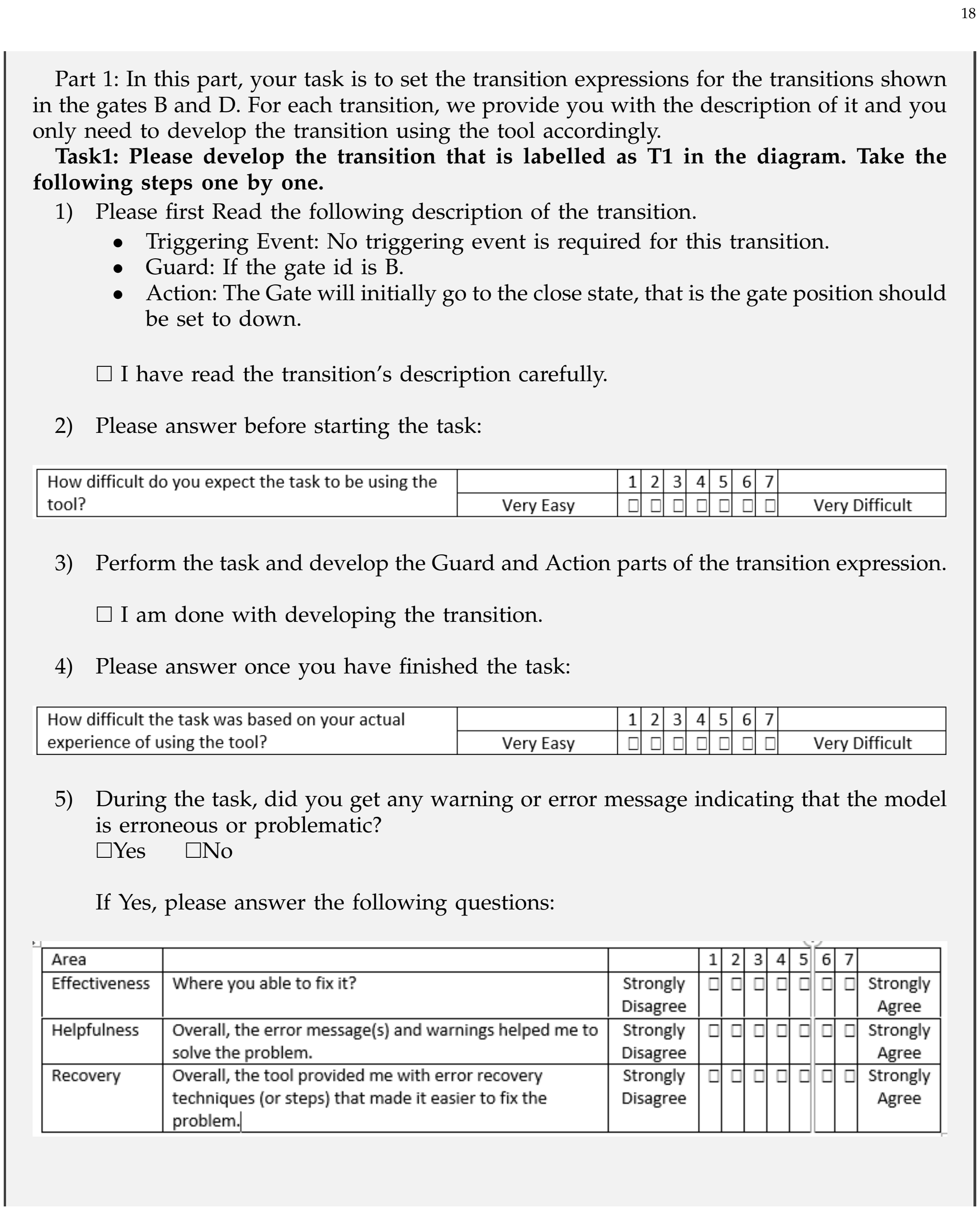

Var g: Gate
Gate A
after (5 secs) [g.blockage = true]
[g.Sensor.car_at_gate = true] / g.position=gatePos.up
[g.id=GateID.A] / g.position=gatePos.dow n
Closed
Open
after (5 secs) [g.Sensor.blockage != true] / g.position=gatePos.dow n
Var g: Gate
Gate B
Controlled Access
after (5 secs) [g.Sensor.blockage = true]
Transponder(g, id) [g.id=GateID.B] / Parking.entringParkingLot(g.Sensor.TransponderID) ; g.position=gatePos.up
T1
Closed
Open
T2
Hold Open
entry / Parking.entringParkingLot(g.Sensor.TransponderID)
at (UndergroundLot.restStartTime) [g.Sensor.blockage != true] / g.position=gatePos.dow n
T3
Var g: Gate
Gate C
after (5 secs) [g.Sensor.blockage = true]
[g.Sensor.car_at_gate = true] / g.position=gatePos.up
[g.id=GateID.C] / g.position=gatePos.dow n
Closed
Open
after (5 secs) [g.Sensor.blockage != true] / g.position=gatePos.dow n
Var g: Gate
Gate D
Transponder(g, id) [g.id=GateID.D AND Parking.isCurrentlyParked(g.Sensor.TransponderID) =true] / Parking.leavingParkingLot(g.Sensor.TransponderID) ; g.position=gatePos.up
Closed
T4
/ g.position=gatePos.up
Open
[g.id=GateID.D] / g.position=gatePos.dow n
after (5 secs) [g.Sensor.blockage != true] / g.position=gatePos.dow n
after (5 secs) [gate.Sensor.blockage = true]

Fig. 2: Partial State Diagram for the Parking Lot application



18

Part 1: In this part, your task is to set the transition expressions for the transitions shown in the gates B and D. For each transition, we provide you with the description of it and you only need to develop the transition using the tool accordingly.

Task1: Please develop the transition that is labelled as T1 in the diagram. Take the following steps one by one.

1) Please first Read the following description of the transition.
   - Triggering Event: No triggering event is required for this transition.
   - Guard: If the gate id is B.
   - Action: The Gate will initially go to the close state, that is the gate position should be set to down.

   ☐ I have read the transition's description carefully.

2) Please answer before starting the task:

How difficult do you expect the task to be using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

3) Perform the task and develop the Guard and Action parts of the transition expression.

   ☐ I am done with developing the transition.

4) Please answer once you have finished the task:

How difficult the task was based on your actual experience of using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

5) During the task, did you get any warning or error message indicating that the model is erroneous or problematic?
   ☐Yes ☐No

   If Yes, please answer the following questions:

Area			1	2	3	4	5	6	7	
Effectiveness	Where you able to fix it?	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Helpfulness	Overall, the error message(s) and warnings helped me to solve the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Recovery	Overall, the tool provided me with error recovery techniques (or steps) that made it easier to fix the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree

Task2: Please develop the transition that is labelled as T2 in the diagram. Take the following steps one by one.

1) Please first Read the following description of the transition (this is a timed transition).
   - Triggering Event: After 5 seconds trigger the transition to check if the gate can be closed.
   - Guard: If the gate is not blocked, that is the gate's sensor does not sense any blockage.
   - Action: Set the gate's position to be closed.

   ☐ I have read the transition's description carefully.

2) Please answer before starting the task:

		1	2	3	4	5	6	7	
How difficult do you expect the task to be using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

3) Perform the task and develop the Event, Guard and Action parts of the transition expression.

   ☐ I am done with developing the transition.

4) Please answer once you have finished the task:

		1	2	3	4	5	6	7	
How difficult the task was based on your actual experience of using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

5) During the task, did you get any warning or error message indicating that the model is erroneous or problematic?
   ☐Yes ☐No

   If Yes, please answer the following questions:

Area			1	2	3	4	5	6	7	
Effectiveness	Where you able to fix it?	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Helpfulness	Overall, the error message(s) and warnings helped me to solve the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Recovery	Overall, the tool provided me with error recovery techniques (or steps) that made it easier to fix the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree

Task3: Please develop the transition that is labelled as T3 in the diagram. Take the following steps one by one.

1) Please first Read the following description of the transition.
   - Triggering Event: At the start of the underground lot's restricted time.
   - Guard: If the gate's sensor senses blockage.
   - Action: Set the gate's position to be open.

   ☐ I have read the transition's description carefully.

2) Please answer before starting the task:

		1	2	3	4	5	6	7	
How difficult do you expect the task to be using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

3) Perform the task and develop the Event, Guard and Action parts of the transition expression.

   ☐ I am done with developing the transition.

4) Please answer once you have finished the task:

		1	2	3	4	5	6	7	
How difficult the task was based on your actual experience of using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

5) During the task, did you get any warning or error message indicating that the model is erroneous or problematic?
   ☐Yes ☐No

   If Yes, please answer the following questions:

Area			1	2	3	4	5	6	7	
Effectiveness	Where you able to fix it?	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Helpfulness	Overall, the error message(s) and warnings helped me to solve the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Recovery	Overall, the tool provided me with error recovery techniques (or steps) that made it easier to fix the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree

Task4: Please develop the transition that is labelled as T4 in the diagram. Take the following steps one by one.

1) Please first Read the following description of the transition.
   - Triggering Event: No triggering event is required for this transition.
   - Guard: The payment received at the gate's Coin Slot is greater than or equal to the gate's Exit Fee.
   - Action: Set the position of the gate to be open.

   ☐ I have read the transition's description carefully.

2) Please answer before starting the task:

		1	2	3	4	5	6	7	
How difficult do you expect the task to be using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

3) Perform the task and develop the Guard part of the transition expression.

   ☐ I am done with developing the transition.

4) Please answer once you have finished the task:

		1	2	3	4	5	6	7	
How difficult the task was based on your actual experience of using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

5) During the task, did you get any warning or error message indicating that the model is erroneous or problematic?
   ☐Yes ☐No

   If Yes, please answer the following questions:

Area			1	2	3	4	5	6	7	
Effectiveness	Where you able to fix it?	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Helpfulness	Overall, the error message(s) and warnings helped me to solve the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Recovery	Overall, the tool provided me with error recovery techniques (or steps) that made it easier to fix the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree

Part 2: Sometimes it is possible that a modeller does not create well-formed, correct and consistent diagrams. It can be because of either not caring about these errors, or not even knowing about their existence. In this part, your task is to modify or observe the model to find inconsistency or error in the model, if any, and fix them.

Task5: Assume that you are supposed to change the name of the gates from A and D to GA and GD respectively.

1) Please answer before starting the task:

		1	2	3	4	5	6	7	
How difficult do you expect the task to be using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

2) Realize the change on the model. Please note that, this change may or may not cause inconsistencies in the model. If you believe that there are inconsistencies, please resolve them appropriately.

   ☐ I am done with changing the model.

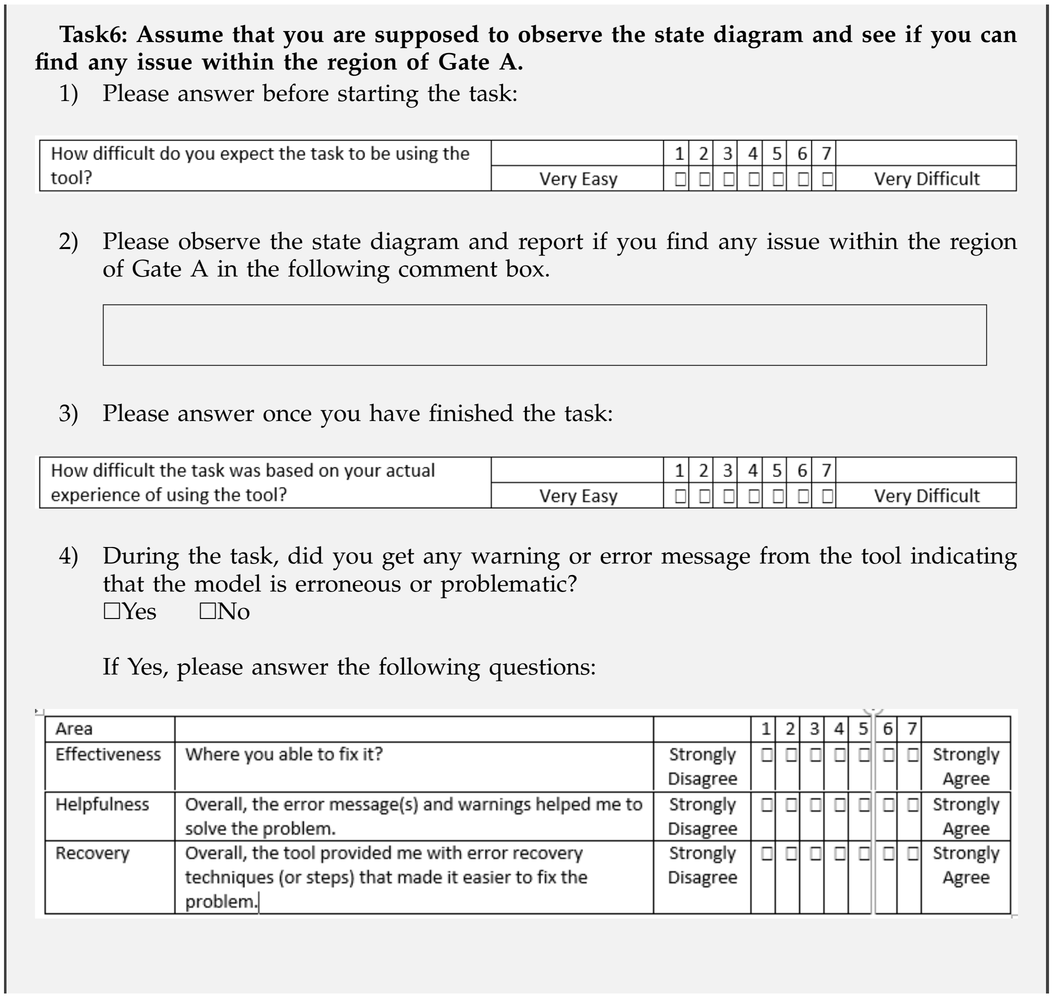
3) Please answer once you have finished the task:

		1	2	3	4	5	6	7	
How difficult the task was based on your actual experience of using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

4) During the task, did you get any warning or error message indicating that the model is erroneous or problematic?
   ☐Yes ☐No

   If Yes, please answer the following questions:

Area			1	2	3	4	5	6	7	
Effectiveness	Where you able to fix it?	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Helpfulness	Overall, the error message(s) and warnings helped me to solve the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Recovery	Overall, the tool provided me with error recovery techniques (or steps) that made it easier to fix the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree



Task6: Assume that you are supposed to observe the state diagram and see if you can find any issue within the region of Gate A.

1) Please answer before starting the task:

		1	2	3	4	5	6	7	
How difficult do you expect the task to be using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

2) Please observe the state diagram and report if you find any issue within the region of Gate A in the following comment box.

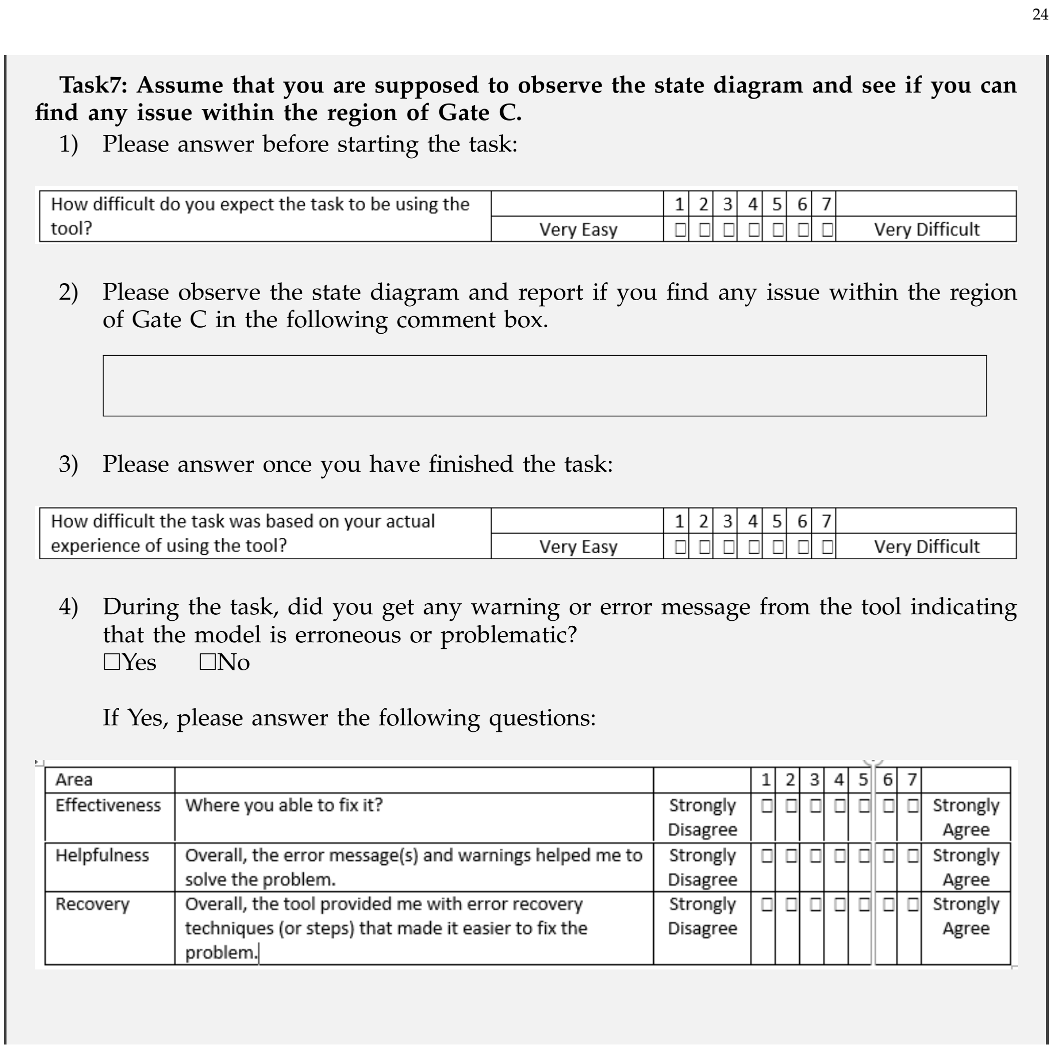
3) Please answer once you have finished the task:

		1	2	3	4	5	6	7	
How difficult the task was based on your actual experience of using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

4) During the task, did you get any warning or error message from the tool indicating that the model is erroneous or problematic?
☐Yes ☐No

If Yes, please answer the following questions:

Area			1	2	3	4	5	6	7	
Effectiveness	Where you able to fix it?	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Helpfulness	Overall, the error message(s) and warnings helped me to solve the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Recovery	Overall, the tool provided me with error recovery techniques (or steps) that made it easier to fix the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree


24

Task7: Assume that you are supposed to observe the state diagram and see if you can find any issue within the region of Gate C.

1) Please answer before starting the task:

		1	2	3	4	5	6	7	
How difficult do you expect the task to be using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

2) Please observe the state diagram and report if you find any issue within the region of Gate C in the following comment box.

3) Please answer once you have finished the task:

		1	2	3	4	5	6	7	
How difficult the task was based on your actual experience of using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

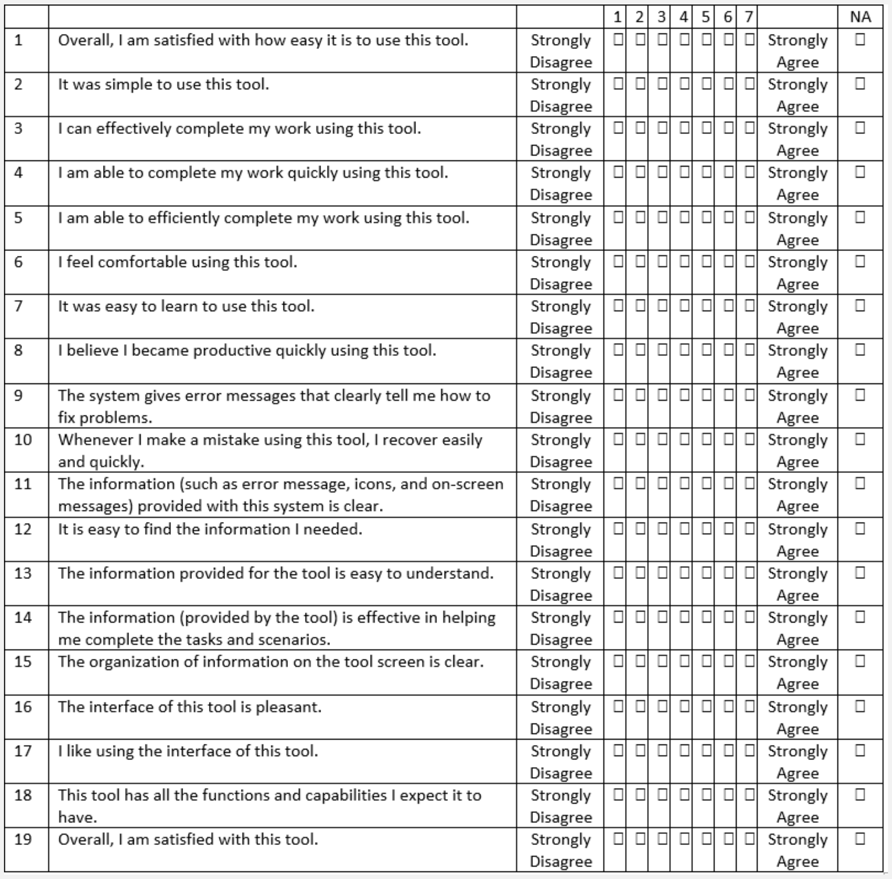
4) During the task, did you get any warning or error message from the tool indicating that the model is erroneous or problematic?
☐Yes ☐No

If Yes, please answer the following questions:

Area			1	2	3	4	5	6	7	
Effectiveness	Where you able to fix it?	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Helpfulness	Overall, the error message(s) and warnings helped me to solve the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Recovery	Overall, the tool provided me with error recovery techniques (or steps) that made it easier to fix the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree

☐ I am done with performing all the tasks.

If you are done with all the tasks, please fill the following task satisfaction form.



			1	2	3	4	5	6	7		NA
1	Overall, I am satisfied with how easy it is to use this tool.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree	☐
2	It was simple to use this tool.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree	☐
3	I can effectively complete my work using this tool.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree	☐
4	I am able to complete my work quickly using this tool.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree	☐
5	I am able to efficiently complete my work using this tool.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree	☐
6	I feel comfortable using this tool.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree	☐
7	It was easy to learn to use this tool.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree	☐
8	I believe I became productive quickly using this tool.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree	☐
9	The system gives error messages that clearly tell me how to fix problems.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree	☐
10	Whenever I make a mistake using this tool, I recover easily and quickly.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree	☐
11	The information (such as error message, icons, and on-screen messages) provided with this system is clear.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree	☐
12	It is easy to find the information I needed.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree	☐
13	The information provided for the tool is easy to understand.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree	☐
14	The information (provided by the tool) is effective in helping me complete the tasks and scenarios.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree	☐
15	The organization of information on the tool screen is clear.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree	☐
16	The interface of this tool is pleasant.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree	☐
17	I like using the interface of this tool.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree	☐
18	This tool has all the functions and capabilities I expect it to have.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree	☐
19	Overall, I am satisfied with this tool.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree	☐

Fig. 3: The CSUQ developed by Lewis.

Thank you for your time and participation in the study.

Appendix D

Materials for the User Studies Conducted to Evaluate Our Tool

Task Description

A modeller is responsible for editing several types of models such as Class Diagram, Sequence Diagram, State Diagram, Activity Diagram, and Use-Case Diagram. Assume that as a modeller, you are given a description of a system to enforce the parking policies at the Bauer complex, and your task is to update the State Diagram. Although your task is only to update/complete the State Diagram, you may find it necessary to also update the existing Class Diagram (i.e. adding a few model elements). You need to set the different parts of the transitions, namely, Triggering Event, Guard or Action, based on the description that is given to you for each part.

(See Next Page)

System Description: You are to create a UML State-Machine model of the proposed system to enforce the parking policies at the Bauer complex. Bauer provides parking for two types of users: customers and employees. There are two levels of parking: surface-level parking and underground parking. Surface-level parking is for customers only. They pay a $2 flat fee for parking. Employees park for free, but only if they park underground. Underground parking is also open to customers in the evenings, but of course customers must still pay ($2) to use it.

Gates control access to both levels of parking. Each gate controls traffic flowing in a single direction (e.g., traffic entering the complex), and it opens only for cars that approach the gate from the appropriate direction. Employees' cars have transponders that the gates can sense. Some gates open only for cars that have transponders. In addition, the system uses transponder data (IDs) to keep track of where employees' cars park. There are four gates that are controlled independently:

1) Gate **A** controls entry into the Bauer complex. It opens whenever it senses a car approaching from outside the Bauer complex. Cars that pass through gate A enter the surface-level parking lot.
2) Gate **B** controls access from surface-level parking to underground parking. Normally, it opens only if an approaching car has a transponder. But during evenings (18:00-6:00), it remains open to all cars.
3) Gate **C** opens whenever it senses a car approaching from the underground parking lot.
4) Gate **D** controls exits from the Bauer complex and collects parking fees. The gate opens whenever a $2 fee is paid (no change is provided). In addition, if an approaching car has a transponder, the gate opens if the car had parked underground (i.e., if the transponder triggered gate B since the car entered the complex).

Each of the gates closes 5 seconds after it opens, unless a blockage is detected, in which case it remains open and tries to close again after another 5 seconds. (Don't worry about modelling the delay of the gate opening and closing.)

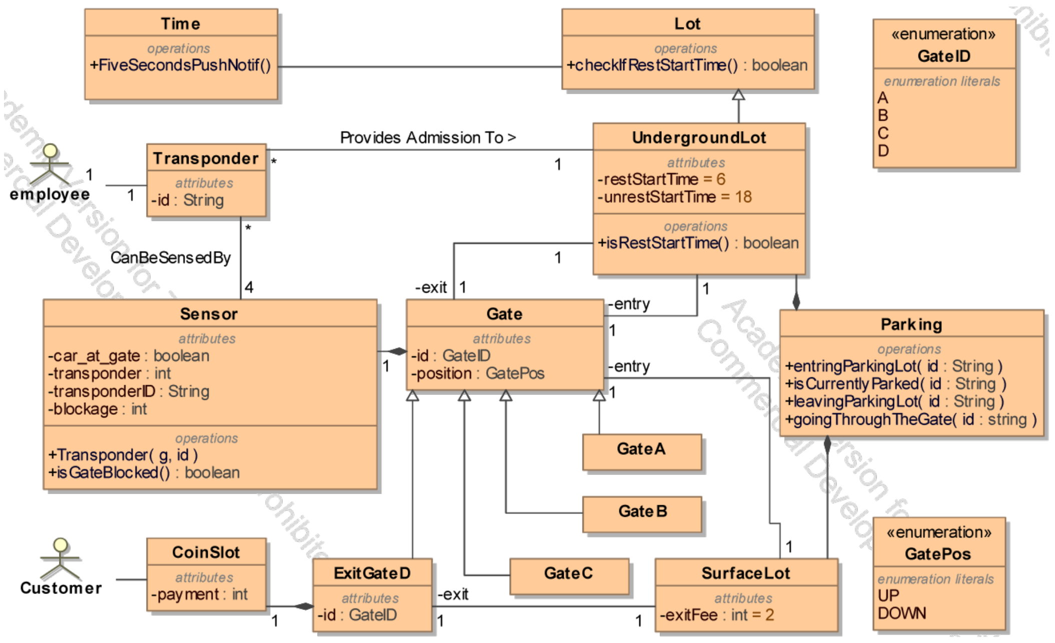


Fig. 1: Class Diagram for the Parking Lot application

Fig. 1 is the partial domain model for the system based on the interface choices already made. The events, conditions, and actions in your state-machine model should be expressed in terms of elements in the domain model. Moreover, the description of the domain model is given below. It helps you understand the domain, classes, attributes and the messages (operations) that you may need to know during the study. It is suggested that you take a brief look at the description before starting the tasks.

Lot: A general class representing the Lot concept.

- *checkIfRestStartTime:* Checks if a lot is in its restricted time period.

Time: Represents the time(er) concept.

- *FiveSecondsPushNotif:* Notifies the lots every five seconds to close the gate.

UndergroundLot: Represents the Underground parking of the parking lot.

- *restStartTime:* The Underground Lot has a restricted time for customers starting at 6:00am.
- *unrestStartTime:* The Underground Lot becomes unrestricted for customers starting at 18:00pm.
- *isRestStartTime:* Checks if the Underground Lot is in its restricted time period.

SurfaceLot: Represents the Surface-Level parking of the parking lot.

- *exitFee:* Represents the $2 parking fee that each customer must pay when exiting the lot.

GateID: Each Gate should have an ID of either A, B, C, or D. GateID is an enum variable (class) that represents the id of each gate.

GatePos: Gates can be either open or close, that is the gates' position can be either up or down. This enum class represents the allowable values for the gates' position.

Gate: Represents the Gates for both underground and surface-level parking lots.
- *id:* Each Gate has an id of the type GateID, that is a Gate can have an id of A, B, C, or D.
- *position:* Each Gate has a position attribute that is from the type of the class GatePos, and can have a value of up or down.

ExitGate: It shows the exit gate of the surface-level parking where the customer needs to put money in the coin slot.

CoinSlot: When exiting from the exit gate of the surface-level parking, the customer needs to pay the exit fee through a coin slot. The CoinSlot class represents the coin slot entity of the domain.
- *payment:* The amount that the customer has already inserted to the coin slot.

Sensor: Each Gate has a sensor the senses if a car is approaching the gate.
- *car_at_gate:* This is a boolean attribute that returns true if there is a car at the gate, and returns false if there is no car at the gate.
- *transponder:* Returns true if an approaching car has a transponder. Otherwise, returns false.
- *transponderID:* If an approaching car has a transponder, this attribute returns the ID of the transponder.
- *blockage:* Returns true in a case that a car has come through the gate, so that the gate does not close.
- *Transponder (g, id):* Returns true if a transponder (with id=id) is approaching gate **g** from the sensed direction.
- *isGateBlocked:* Returns true if the Gate is blocked.

Transponder: Each employee has a Transponder installed on their car. that provides admission to the underground parking and can be sensed by the sensors on the gates.
- *id:* Each Transponder has an ID.

Parking: This is a singleton class which records the states of cars with the transponder. Moreover, if a car enters to the parking, its transponder id should be recorded and be added to the collection of the parked cars' transponder ids. Also, when it leaves the parking, the car's transponder id must be removed from the list.
- *enteringParkingLot (id):* When a car is entering to the parking lot, this operation adds the transponder id of the car to the list of the transponder ids of the existing cars in the parking.
- *isCurrentlyParked (id):* This is a boolean operation which checks if a car with a transponder id had parked in the parking.
- *leavingParkingLot (id):* When a car is leaving the parking, the system uses this operation to remove the transponder id of the leaving car from the list of the transponder ids of the parked cars.
- *goingThroughTheGate (id):* Same as the enteringParkingLot operation.

1 Tasks for Context Study

Task1: Open the State Machine for the Gate B. Find the transition that is labelled as Tran1.

1) Please first Read the following description of the transition.
   - Event: No triggering event is required for this transition.
   - Guard: If the gate id is B.
   - Action: The gate should initially become closed; that is, the gate's position should be set to down.

   ☐ I have read the task's description carefully.
2) Please answer before starting the task:

		1	2	3	4	5	6	7	
How difficult do you expect the task to be using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

3) Perform the task and develop the Guard and Action parts of the transition expression.

   ☐ I am done with developing the transition.
4) Please answer once you have finished the task:

		1	2	3	4	5	6	7	
How difficult the task was based on your actual experience of using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

Task2: Open the State Machine for the Gate B. Find the transition that is labelled as Tran2.

1) Please first Read the following description of the transition.
   - Event: No triggering event is required for this transition.
   - Guard: If the time is NOT during the underground lots restricted time and the gate's sensor senses blockage.
   - Action: The gate should become open; that is, the gates position should be set to up.

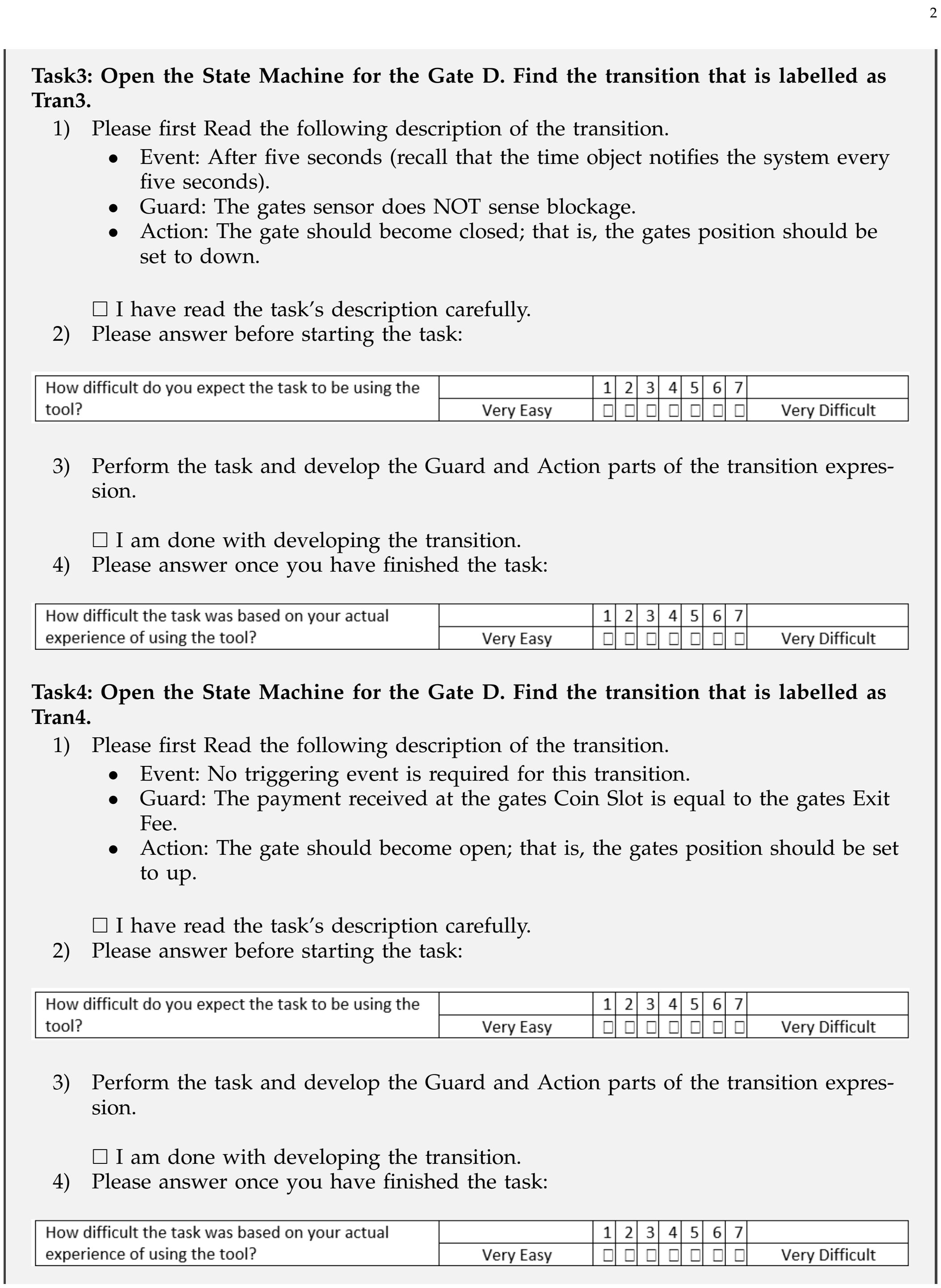
   ☐ I have read the task's description carefully.
2) Please answer before starting the task:

		1	2	3	4	5	6	7	
How difficult do you expect the task to be using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

3) Perform the task and develop the Guard and Action parts of the transition expression.

   ☐ I am done with developing the transition.
4) Please answer once you have finished the task:

		1	2	3	4	5	6	7	
How difficult the task was based on your actual experience of using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult



Task3: Open the State Machine for the Gate D. Find the transition that is labelled as Tran3.

1) Please first Read the following description of the transition.
   - Event: After five seconds (recall that the time object notifies the system every five seconds).
   - Guard: The gates sensor does NOT sense blockage.
   - Action: The gate should become closed; that is, the gates position should be set to down.

   ☐ I have read the task's description carefully.
2) Please answer before starting the task:

How difficult do you expect the task to be using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

3) Perform the task and develop the Guard and Action parts of the transition expression.

   ☐ I am done with developing the transition.
4) Please answer once you have finished the task:

How difficult the task was based on your actual experience of using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

Task4: Open the State Machine for the Gate D. Find the transition that is labelled as Tran4.

1) Please first Read the following description of the transition.
   - Event: No triggering event is required for this transition.
   - Guard: The payment received at the gates Coin Slot is equal to the gates Exit Fee.
   - Action: The gate should become open; that is, the gates position should be set to up.

   ☐ I have read the task's description carefully.
2) Please answer before starting the task:

How difficult do you expect the task to be using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

3) Perform the task and develop the Guard and Action parts of the transition expression.

   ☐ I am done with developing the transition.
4) Please answer once you have finished the task:

How difficult the task was based on your actual experience of using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

Task5: Open the State Machine for the Gate D. Find the transition that is labelled as Tran5.

1) Please first Read the following description of the transition.
   - Event: No triggering event is required for this transition.
   - Guard: if the gate is blocked.
   - Action: No action is required for this transition.

   ☐ I have read the task's description carefully.
2) Please answer before starting the task:

How difficult do you expect the task to be using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

3) Perform the task and develop the Guard and Action parts of the transition expression.

   ☐ I am done with developing the transition.
4) Please answer once you have finished the task:

How difficult the task was based on your actual experience of using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

Task6: Open the State Machine for the Gate C. Find the transition that is labelled as Tran6.

1) Please first Read the following description of the transition.
   - Event: Similar as T6Copy in GateA..
   - Guard: Similar as T6Copy in GateA.
   - Action: Similar as T6Copy in GateA.

   ☐ I have read the task's description carefully.
2) Please answer before starting the task:

How difficult do you expect the task to be using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

3) Perform the task and develop the Guard and Action parts of the transition expression.

   ☐ I am done with developing the transition.
4) Please answer once you have finished the task:

How difficult the task was based on your actual experience of using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

Task7: Open the State Machine for the Gate B. Find the transition that is labelled as Tran7.

1) Please first Read the following description of the transition.
    - Event: No triggering event is required for this transition.
    - Guard: No guard is required for this transition.
    - Action: Create a new class named GateCounter with a new function named as increaseCounter and then use it. Also, create an association between the GateCounter class and GateB class.

    ☐ I have read the task's description carefully.
2) Please answer before starting the task:

How difficult do you expect the task to be using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

3) Perform the task and develop the Guard and Action parts of the transition expression.

    ☐ I am done with developing the transition.
4) Please answer once you have finished the task:

How difficult the task was based on your actual experience of using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

Task8: Find the transition that is labelled as Tran7 in Gate B and the transition that is labelled as Tran8 in Gate D.

1) Please first Read the following description of the transition.
    - Event: Use a function named as Transponder. You may need to create this function in the Sensor class before setting it in the Event.
    - Guard: No guard is required for this transition.
    - Action: No action is required for this transition.

    ☐ I have read the task's description carefully.
2) Please answer before starting the task:

How difficult do you expect the task to be using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

3) Perform the task and develop the Guard and Action parts of the transition expression.

    ☐ I am done with developing the transition.
4) Please answer once you have finished the task:

How difficult the task was based on your actual experience of using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

Task9: In this task, you need to extend the Class diagram to cover a new gate, namely Gate E, which controls the second entry into the Bauer complex for persons with disabilities and who have access to dedicated accessible-parking spots. You also need to develop a new State-Machine diagram for the gate E. Please edit the Class diagram and the State-Machine diagram based on the description given below.

1) Please first Read the following description of the transition.
   A disability-car that has an occupant with disabilities must have a valid permit card with an id and expiry date on it). When approaching the Gate E, the card scanner machine asks him/her to tap the permit card. The card scanner machine checks the card's validity (i.e., the card's id is registered in the parking lot's database, and the expiry date of the card is greater than or equal to the current date) and opens the gate if the card is valid. After 5 seconds, the system attempts to close the gate only if the gate is not blocked.

   ☐ I have read the task's description carefully.
2) Please answer before starting the task:

How difficult do you expect the task to be using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

3) Perform the task and develop the Guard and Action parts of the transition expression.

   ☐ I am done with developing the transition.
4) Please answer once you have finished the task:

How difficult the task was based on your actual experience of using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

2 Post-Session Experience Ratings for Context Study

Please answer to the following questions based on your actual experience using the tool in the study.

1) Compared to your initial expectation, To what extent did the tool was close to satisfy your expectation?

   Below My Expectation ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 Above My Expectation

2) The Contextual User Interfaces capabilities meet my requirements.

   Strongly Disagree ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 Strongly Agree

3) The Contextual User Interface is easy to use.

   Strongly Disagree ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 Strongly Agree

4) During performing the tasks, to what extent did you experience the following challenges?
   - Relying on memory (Remembering the contextual information relevant to performing the tasks).

     Strongly Disagree ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 Strongly Agree

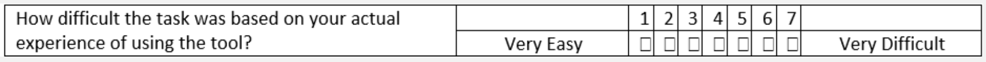
   - Switching between artifacts (Searching for information) and changing the focus.

     Strongly Disagree ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 Strongly Agree

   - Maintaining the overview of the diagrams.

     Strongly Disagree ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 Strongly Agree

   - Too much scrolling and zooming.



How difficult the task was based on your actual experience of using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

   - Knowing the relations between artifacts.

     Strongly Disagree ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 Strongly Agree

   - Arranging windows or Finding the right window.

     Strongly Disagree ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 Strongly Agree

Thank you for your time and participation in the study.

3 Pre-Session Expectation Ratings for Debugging Study

Please answer to the following questions based on your overall expectations of using modelling tools.

1) To what extent do you agree that inconsistencies should be prevented immediately rather than allowing them to exist and fix them at a later time?

   Very Easy ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 Very Difficult

2) To what extent do you agree that maintaining inconsistencies at all time is counter-productive and intrusive (disrupting the focus tasks)?

   Very Easy ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 Very Difficult

4 Tasks for Debugging Study

Task1: Find the GateID enumeration class. Rename the class name from GateID to GateIDs. If you believe that this change will introduce inconsistencies, please find them and fix them accordingly.

1) ☐ I have read the task's description carefully.
2) Please answer before starting the task:

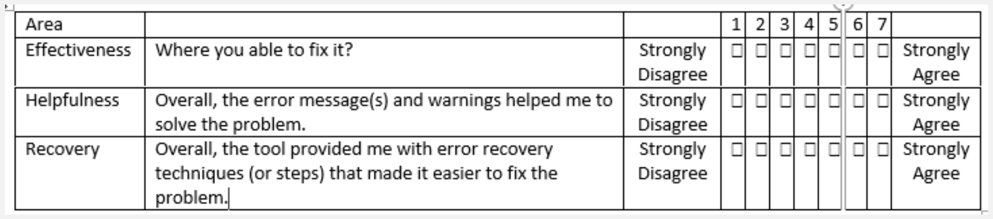
How difficult do you expect the task to be using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

3) Perform the task.
   ☐ I am done with developing the transition.
4) Please answer once you have finished the task:

How difficult the task was based on your actual experience of using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

5) During the task, did you get any warning or error message from the tool indicating that the model is erroneous or problematic?
   ☐Yes ☐No

   If Yes, please answer the following questions:



Area			1	2	3	4	5	6	7	
Effectiveness	Where you able to fix it?	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Helpfulness	Overall, the error message(s) and warnings helped me to solve the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Recovery	Overall, the tool provided me with error recovery techniques (or steps) that made it easier to fix the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree

Task2: The class Gate is a superclass of ExitGateD class. Gate and ExitGateD both have an attribute named as id. You, as a modeller find that the id attribute in ExitGateD is redundant and should be removed from the ExitGateD class. Remove it and also fix the inconsistencies that this task may introduce to the model, if any.

1) ☐ I have read the task's description carefully.
2) Please answer before starting the task:

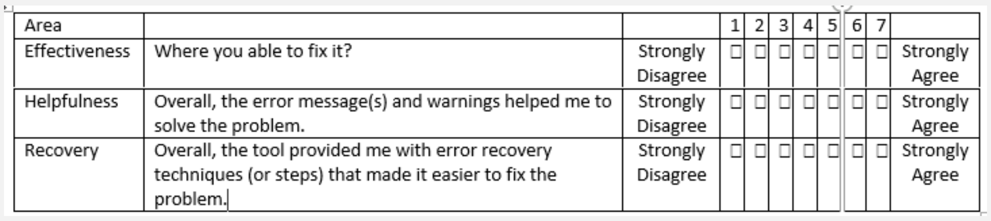
How difficult do you expect the task to be using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

3) Perform the task.
   ☐ I am done with developing the transition.
4) Please answer once you have finished the task:

How difficult the task was based on your actual experience of using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

5) During the task, did you get any warning or error message from the tool indicating that the model is erroneous or problematic?
☐Yes ☐No

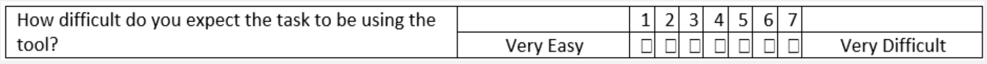
If Yes, please answer the following questions:



Area			1	2	3	4	5	6	7	
Effectiveness	Where you able to fix it?	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Helpfulness	Overall, the error message(s) and warnings helped me to solve the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Recovery	Overall, the tool provided me with error recovery techniques (or steps) that made it easier to fix the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree

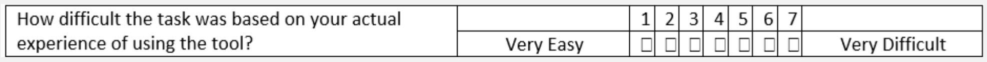
Task3: Find class Gate and rename the position attribute to pos. Fix the inconsistencies, if any, that this task may introduce to the model.

1) ☐ I have read the task's description carefully.
2) Please answer before starting the task:



		1	2	3	4	5	6	7	
How difficult do you expect the task to be using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

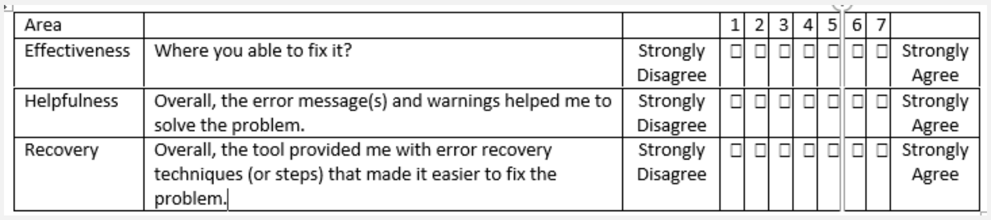
3) Perform the task.
☐ I am done with developing the transition.
4) Please answer once you have finished the task:



		1	2	3	4	5	6	7	
How difficult the task was based on your actual experience of using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

5) During the task, did you get any warning or error message from the tool indicating that the model is erroneous or problematic?
☐Yes ☐No

If Yes, please answer the following questions:



Area			1	2	3	4	5	6	7	
Effectiveness	Where you able to fix it?	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Helpfulness	Overall, the error message(s) and warnings helped me to solve the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Recovery	Overall, the tool provided me with error recovery techniques (or steps) that made it easier to fix the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree

Task4: The Parking class has an operation named as enteringParkingLot. You, as a modeller find that the enteringParkingLot is a redundant operation and should be

removed from the Parking class. Remove it and also fix all the transitions that are referring to the enteringParkingLot as explained below:

- Event: If enteringParkingLot is used in the event, change the event to goingThroughTheGate.
- Guard: If enteringParkingLot is used in the guard, delete the guard. Consequently, if the containing transition will have no event, guard, or action after deleting the guard, then delete the whole transition.
- Action: If enteringParkingLot is used in the action/effect, delete the action. Consequently, if the containing transition will have no event, guard, or action after deleting the action, then delete the whole transition.

1) ☐ I have read the task's description carefully.
2) Please answer before starting the task:

How difficult do you expect the task to be using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

3) Perform the task.
   ☐ I am done with developing the transition.
4) Please answer once you have finished the task:

How difficult the task was based on your actual experience of using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

5) During the task, did you get any warning or error message from the tool indicating that the model is erroneous or problematic?
   ☐Yes ☐No

   If Yes, please answer the following questions:

Area			1	2	3	4	5	6	7	
Effectiveness	Where you able to fix it?	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Helpfulness	Overall, the error message(s) and warnings helped me to solve the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Recovery	Overall, the tool provided me with error recovery techniques (or steps) that made it easier to fix the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree

Task5: The Sensor class has the Blockage attribute with the type of int. Change the Blockage attributes type from int to bool and find any inconsistencies it may occur.

1) ☐ I have read the task's description carefully.
2) Please answer before starting the task:

How difficult do you expect the task to be using the tool?		1	2	3	4	5	6	7	
	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

3) Perform the task.

☐ I am done with developing the transition.

4) Please answer once you have finished the task:

		1	2	3	4	5	6	7	
How difficult the task was based on your actual experience of using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

5) During the task, did you get any warning or error message from the tool indicating that the model is erroneous or problematic?
☐Yes ☐No

If Yes, please answer the following questions:

Area			1	2	3	4	5	6	7	
Effectiveness	Where you able to fix it?	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Helpfulness	Overall, the error message(s) and warnings helped me to solve the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Recovery	Overall, the tool provided me with error recovery techniques (or steps) that made it easier to fix the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree

Task6: The Sensor class has the car_ at_ gate attribute with the type of bool. Change the car_ at_ gate attributes type from bool to int and find any inconsistencies it may occur.

1) ☐ I have read the task's description carefully.
2) Please answer before starting the task:

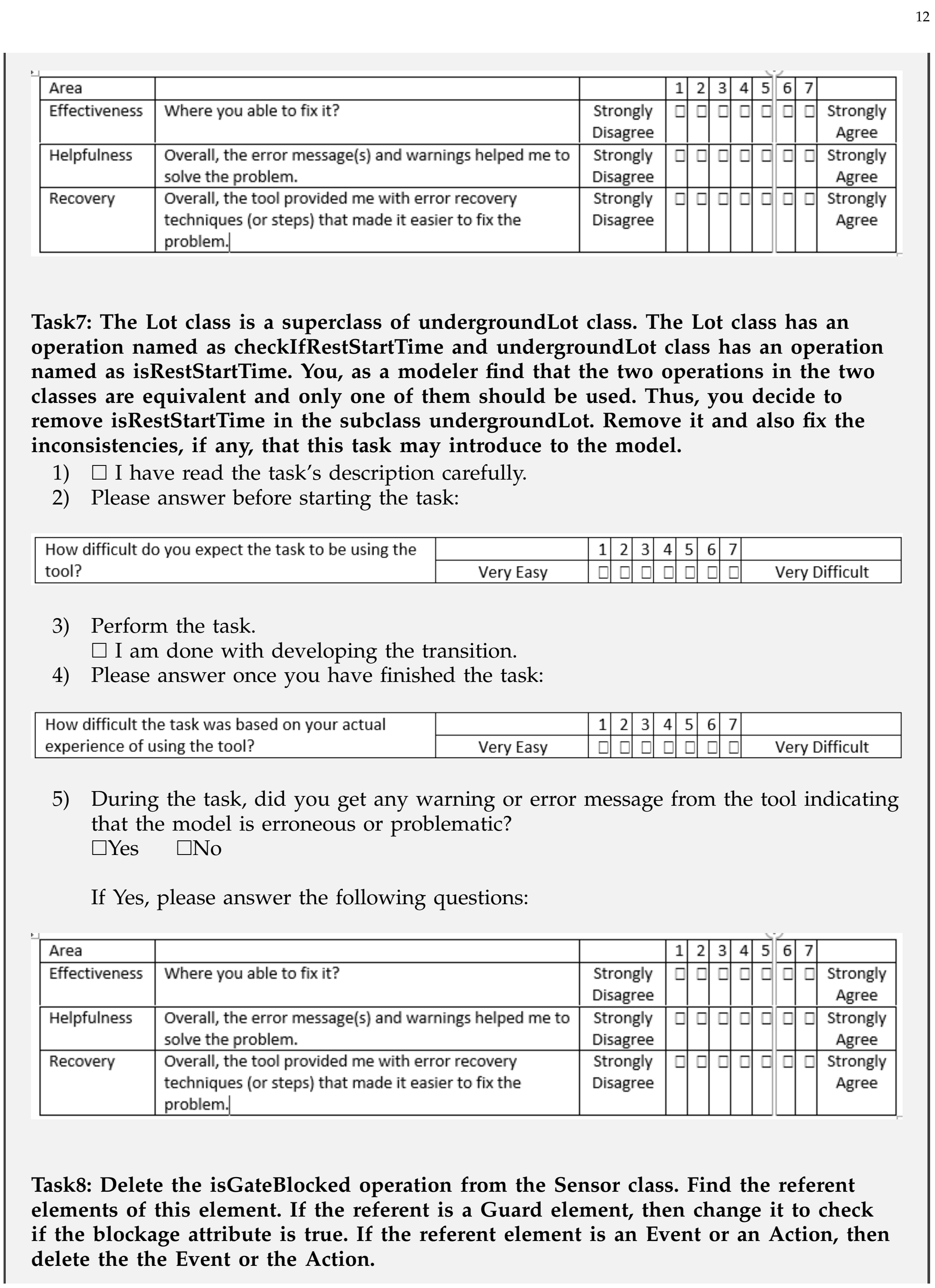
		1	2	3	4	5	6	7	
How difficult do you expect the task to be using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

3) Perform the task.
☐ I am done with developing the transition.
4) Please answer once you have finished the task:

		1	2	3	4	5	6	7	
How difficult the task was based on your actual experience of using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

5) During the task, did you get any warning or error message from the tool indicating that the model is erroneous or problematic?
☐Yes ☐No

If Yes, please answer the following questions:



Area			1	2	3	4	5	6	7	
Effectiveness	Where you able to fix it?	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Helpfulness	Overall, the error message(s) and warnings helped me to solve the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Recovery	Overall, the tool provided me with error recovery techniques (or steps) that made it easier to fix the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree

Task7: The Lot class is a superclass of undergroundLot class. The Lot class has an operation named as checkIfRestStartTime and undergroundLot class has an operation named as isRestStartTime. You, as a modeler find that the two operations in the two classes are equivalent and only one of them should be used. Thus, you decide to remove isRestStartTime in the subclass undergroundLot. Remove it and also fix the inconsistencies, if any, that this task may introduce to the model.

1) ☐ I have read the task's description carefully.
2) Please answer before starting the task:

		1	2	3	4	5	6	7	
How difficult do you expect the task to be using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

3) Perform the task.
   ☐ I am done with developing the transition.
4) Please answer once you have finished the task:

		1	2	3	4	5	6	7	
How difficult the task was based on your actual experience of using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

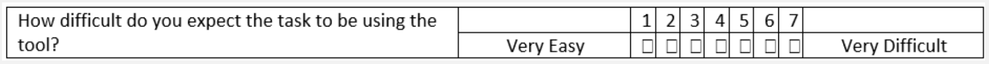
5) During the task, did you get any warning or error message from the tool indicating that the model is erroneous or problematic?
   ☐Yes ☐No

   If Yes, please answer the following questions:

Area			1	2	3	4	5	6	7	
Effectiveness	Where you able to fix it?	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Helpfulness	Overall, the error message(s) and warnings helped me to solve the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Recovery	Overall, the tool provided me with error recovery techniques (or steps) that made it easier to fix the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree

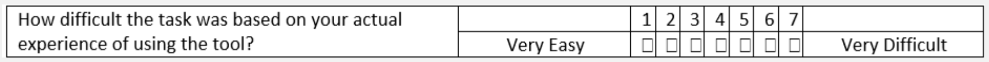
Task8: Delete the isGateBlocked operation from the Sensor class. Find the referent elements of this element. If the referent is a Guard element, then change it to check if the blockage attribute is true. If the referent element is an Event or an Action, then delete the the Event or the Action.

1) ☐ I have read the task's description carefully.
2) Please answer before starting the task:



		1	2	3	4	5	6	7	
How difficult do you expect the task to be using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

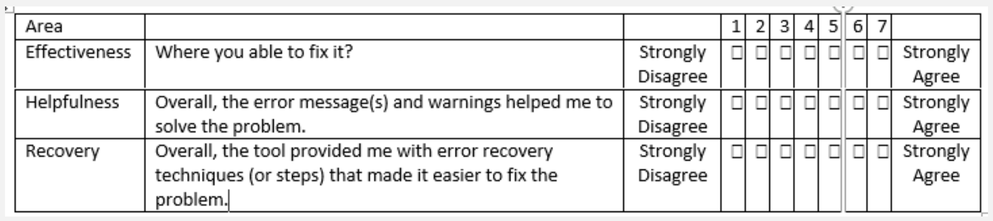
3) Perform the task.
   ☐ I am done with developing the transition.
4) Please answer once you have finished the task:



		1	2	3	4	5	6	7	
How difficult the task was based on your actual experience of using the tool?	Very Easy	☐	☐	☐	☐	☐	☐	☐	Very Difficult

5) During the task, did you get any warning or error message from the tool indicating that the model is erroneous or problematic?
   ☐Yes ☐No

   If Yes, please answer the following questions:



Area			1	2	3	4	5	6	7	
Effectiveness	Where you able to fix it?	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Helpfulness	Overall, the error message(s) and warnings helped me to solve the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree
Recovery	Overall, the tool provided me with error recovery techniques (or steps) that made it easier to fix the problem.	Strongly Disagree	☐	☐	☐	☐	☐	☐	☐	Strongly Agree

5 Post-Session Experience Ratings for Debugging Study

Please answer to the following questions based on your actual experience using the tool in the study.

1) Compared to your initial expectation, To what extent did the tool was close to satisfy your expectation?

   Below My Expectation ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 Above My Expectation

2) The Interactive Consistency Management Interfaces capabilities meet my requirements.

   Strongly Disagree ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 Strongly Agree

3) The Interactive Consistency Management Interface is easy to use.

   Strongly Disagree ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 Strongly Agree

4) How confident you are with your level of success in fulfilling the tasks (do you think you might have left some uncaught errors in the model)?

   Strongly Unconfident ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 Strongly Confident

5) Based on your experience, to what extent do you feel being disrupted by the tool?

   Strongly Disagree ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 Strongly Agree

6) Based on your experience, to what extent do you agree that being disrupted was worth of delivering more correct models and being more confident in the solution?

   Strongly Disagree ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 Strongly Agree

Thank you for your time and participation in the study.