FAKULTÄT FÜR INFORMATIK
DER TECHNISCHE UNIVERSITÄT MÜNCHEN

Forschungs- und Lehreinheit I
Angewandte Softwaretechnik

MiNT: MULTIMODAL iNTERACTION FOR MODELING AND MODEL REFACTORING

Nitesh Narayan

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Nassir Navab, Ph.D.
Prüfer der Dissertation:	1. Univ.-Prof. Bernd Bruegge, Ph.D.
	2. Univ.-Prof. Kirill Krinkin, Ph.D. Saint-Petersburg Electrotechnical University

Die Dissertation wurde am 30.01.2017 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 08.03.2017 angenommen.

Dedicated to my father

Acknowledgements

I would like to thank my adviser, Professor Bernd Bruegge for his continuous guidance, support and critical feedback throughout my research. Thank you so much for providing an amazing atmosphere and numerous opportunities to learn and improve. As a great mentor, you inspired me not only professionally but also as a person. I am sure, I still have a lot to learn from you. Special thanks to Professor Kirill Krinkin, for accepting to be my second supervisor. I got to know you from the course of JASS 2016, an intercultural student's project. It was a great experience and we have had a lot of fun working with the students from St. Petersburg. Thank you for providing such an exclusive opportunity.

I take this opportunity to thank all the colleagues from the Chair for Applied Software Engineering for their continuous support throughout these years. Dr. Jonas Helming and Dr. Maximilian Kögel played a key role during the early days of my work by providing their valuable advice and introducing to the scientific work. Dr. Walid Maalej, and Dr. Dennis Pagano enriched my understanding of scientific work and methodologies through numerous interactive discussions.

Thank you Yang Li! As an amazing friend and colleague you have always been all ears discussing even the silliest of my ideas, and even collaborating on them for field-testing. Thank you Hoda Naguib, for your great support, care, and affection. It is so much fun and friendly ambiance having you around. Thank you Emitzá Guzmán, for all the interesting and insightful conversations we had over time about work and life. Thank you Jan Knobloch and Stefan Nosović. It was always exciting to discuss all the cool happenings on various topics with you guys. Best wishes for your doctoral work.

I would like to express my gratitude to Frau Markl, for accommodating appointments and helping with all the organizational issues at work. Thank you Frau Schneider for keeping me up-to-date with the technical gadgets and instruments during my research work. Thank you Frau Weber for handling the financial issues behind the curtain. A big thank to all the modelers, experiment participants and reviewers who provided their valuable insights during the course of this dissertation. And my best wishes to all the current and prospective doctoral students at the chair.

I am indebted towards my parents and siblings for their continuous care, support, and guidance. You are the reason behind what I am today and I take this as an opportunity to try thanking you. Finally, thank you so much Suchi, for having faith in me and being there through my ups and downs of my life for the last few years (for 11 years as girl-friend and the last two as wife). As a pillar of support you held me strong through the turbulent times with your constant support and understanding. Thank you!

Abstract

The development of software brings together participants from different backgrounds, such as domain experts, analysts, designers, programmers, managers, technical writers, graphic designers, and users. No single participant can understand or control all aspects of the system under development, and thus, all participants depend on others to accomplish their work. Moreover, any change in the system or the application domain requires all participants to update their understanding of the system. The importance of continuous involvement of domain experts in the modeling process is well known. But domain experts are usually not proficient with the modeling tools used by the software developers and as a result are often limited to the initial requirements elicitation. Researchers have provided substantial evidence that multimodal interfaces can greatly expand the accessibility of interfaces to diverse and nonspecialist users.

To address these limitations in the collaboration between application domain experts and modelers, we developed MiNT, an extensible platform to add new modalities and to configure multimodal fusion in CASE tools. MiNT is based on the M^3 framework that allows capturing multimodal interaction during the design process of new multimodal interfaces. The M^3 framework has been developed in a bootstrapping process during the development of MiNT. The viability of MiNT was demonstrated in two reference implementations; Mint Eclipse and Mint Mobile. MiNT Eclipse used the MiNT framework to add multimodality to Eclipse-based modeling. MiNT Mobile provides multimodal modeling and model transformations on mobile devices.

We conducted two controlled experiments to study the feasibility and applicability of multimodal interfaces for modeling and model refactoring. The results of the first experiment show that multimodal interfaces employing speech as an input modality improve the efficiency of modelers. Speech additionally allows modelers to verbalize their thoughts and is suitable for collaborative modeling sessions. The results of the second experiment show that a multimodal interface which provides a combination of touch, speech, and touch gestures is more useful than a multimodal interface employing only touch and speech.

Table of contents

List of figures 13

List of tables 15

1 Introduction 1

1.1 Problem . . . 3

1.2 Research Approach . . . 6

1.3 Outline of the Dissertation . . . 7

2 Foundations 9

2.1 Modeling in Software Engineering . . . 10

2.1.1 Model Transformation . . . 11

2.2 Collaborative Modeling . . . 12

2.3 Multimodal User Interface . . . 16

2.4 Usability Engineering . . . 19

3 Multimodal modeling (M^3) Framework 21

3.1 M^{3}framework . . . 21

3.1.1 Generic Multi-modality Model . . . 22

3.1.2 Use case Meta-model . . . 25

3.2 Models for Visual Modeling Tools . . . 26

3.2.1 Diagram Presentation Modality . . . 26

3.3 Application of M ^{3} framework . . . 27

4 Requirements Specification and Design Goals for Multimodal Modeling 29

4.1 Speech for UML Modeling . . . 29

4.2 Gestures for UML Modeling . . . 32

4.2.1 Results . . . 33

4.3 Requirements for Multimodal Modeling 35
4.3.1 Visionary Scenarios 36
4.3.2 Requirements 37

5 MiNT Framework and Applications 39
5.1 Analysis Model 39
5.2 Design Goals 41
5.3 Architecture of MiNT 43
5.4 Multimodal Integration 45
5.5 MiNT Eclipse 47
5.6 MiNT Mobile 50
5.6.1 MiNT Mobile User Interface 52
5.6.2 Continuous Speech Mode 54

6 Evaluation Multimodal Modeling on Interactive Whiteboard 59
6.1 Context 59
6.2 Variables 61
6.3 Subjects 63
6.4 Setup 63
6.5 Procedure 64
6.6 Experiment Results 65
6.6.1 Collected Data 65
6.6.2 Analysis 67
6.6.3 Exit Interview 69
6.7 Discussion 71
6.8 Threats to Validity 73

7 Evaluation Multimodal Modeling on Mobile Devices 75
7.1 Context 75
7.2 Variables 76
7.3 Subjects 77
7.4 Setup 77
7.5 Procedure 77
7.6 Experiment Results 78
7.6.1 Collected Data 78
7.6.2 Analysis 80
7.6.3 Subjective Evaluation 81

7.7 Discussion 88
7.8 Threats to Validity 88

8 Conclusion and Future Work 91
8.1 Contributions 92
8.2 Limitations 93
8.3 Future Work 93

Appendix A Model Refactoring Taxonomy 95

Appendix B Controlled Experiment 105
B.1 Instructors Checklist 106
B.2 Experiment Task 108
B.3 Subjective Questionnaire 110
B.4 Stastical Analysis: F-test for variance equality 111
B.5 Stastical Analysis: t-Test for unequal variances 113

Appendix C Controlled Experiment 117
C.1 Instructors Checklist 117
C.2 Experiment Task 120
C.3 Subjective Questionnaire 122
C.4 Statistical Analysis: t-Test paired sample 125

References 129

List of figures

2.1 Traditional classification of models 13
2.2 Semi-formal model for multimodal interaction 14
2.3 State diagram showing the transition betwen different states of a model . . 15
2.4 Input and output modality. Adapted from [OS04] 17
2.5 A representation of multimodal man machine interaction loop from [DLO09] 18

3.1 Excerpt from the generic multi-modality model (UML Class diagram) . . . 23
3.2 Input modality model (UML Class diagram) 23
3.3 Output modality model (UML Class diagram) 24
3.4 Unified meta-model of M^3 framework (UML Class diagram) 25
3.5 Diagram presentation modality for the **open diagram** use case (UML Class diagram) 26

4.1 Vocabulary for UML modeling tools 30
4.2 JSGF grammar for speech recognition of simple spoken commands 31

5.1 Object model of interaction modalities for multimodal interaction (UML class diagram) 40
5.2 Object model of interaction definition (UML class diagram) 41
5.3 MiNT framework architecture (UML component diagram) 42
5.4 Example modality integration definition for MiNT framework 44
5.5 Architecture of MiNT Eclipse reference implementation (UML component diagram) 48
5.6 Deployment diagram of MiNT Eclipse (UML deployment diagram) 49
5.7 Multimodal interaction using interactive whiteboard and speech 50
5.8 Architecture of MiNT Mobile reference implementation (UML component diagram) 51
5.9 Deployment diagram of MiNT Mobile (UML deployment diagram) 51

5.10 MiNT Mobile user interface 52
5.11 MiNT Mobile: Informal modeling using touch sketches 53

6.1 Results: Task difficulty vs difficulty understanding the material 70
6.2 Results: Difficulty performing task using tool interface 72

7.1 Educational background and modeling experience of the subjects 79
7.2 Evaluation of pragmatic qualities for touch and speech interface against touch, speech (MiNT TS) and touch gesture interface (MiNT TSG) 82
7.3 Evaluation of hedonic qualities for touch and speech interface against touch, speech (MiNT TS) and touch gesture interface (MiNT TSG) 83
7.4 Evaluation of attractiveness qualities for touch and speech interface against touch, speech (MiNT TS) and touch gesture interface (MiNT TSG) 84
7.5 Evaluation of suitable working contexts for touch and speech interface against touch, speech (MiNT TS) and touch gesture interface (MiNT TSG) 85
7.6 Difficulty performing tasks using touch and speech interface against touch, speech (MiNT TS) and touch gesture interface (MiNT TSG) 86
7.7 Summative evaluation of touch and speech interface against touch, speech (MiNT TS) and touch gesture interface (MiNT TSG) 87

A.1 Pull up attribute use case 97
A.2 Pull up method use case 98
A.3 Extract superclass use case 99
A.4 Extract subclass use case 101
A.5 Attribute to object use case 102
A.6 Collapse hierarchy use case 103
A.7 Merge classes use case 104

B.1 Sample solution object model for Task 1 108
B.2 Existing object model for Task 2 115

C.1 Sample solution object model for Task 1 119
C.2 Object model for Task 2 119

List of tables

3.1 Event flow in the open diagram use case 26
3.2 Steps in create class use case use case 27

4.1 InteractionActions with touch gestures for modeling and model refactoring use cases 34

5.1 Properties to define interaction definition (unimodal or multimodal) with MiNT Eclipse 46
5.2 InteractionActions for Papyrus UML Class diagram editor with touch and speech input 55
5.3 InteractionActions for model refactoring actions 56

6.1 Task Categorization 63
6.2 Collected data from the experiment (part 1) 65
6.3 Collected data from the experiment (part 2) 66
6.4 F-test for the equality of variances 67

7.1 Categorization 77
7.2 Time to complete modeling (task 1), and model transformation (task 2) by subjects 79

Chapter 1

Introduction

Models play an important role in the disciplines of design and engineering. They serve as a basis for design and are easily understandable by non-software engineers [Moo09]. As an important method for people to understand complex systems and to navigate around structured information, models facilitate reasoning, collaboration, communication, and solving a complex problem using the higher level of abstractions [AF03] [BD10]. The history of visual modeling notations in Software Engineering (SE) dates back to the 1940s, with the development of Goldstine and Neumann's program flow chart [NG47]. Subsequently, several other graphical modeling languages followed over time such as Object-Modeling Technique (OMT) [RBP^{+}91], Object-Oriented Software Engineering (OOSE) [Jac93], and eventually Unified Modeling Language (UML) [OMGb] and the Systems Modeling Language (SysML) [OMGa] under the umbrella of Object Management Group (OMG).

Along with the development of modeling languages, Computer Aided Software Engineering tools (CASE-tools) gained prominence among practitioners to enable express designs using graphical representations such as state machines, structure diagrams, and data flow diagrams throughout the software development lifecycle [Sch06]. CASE-tools facilitated construction and manipulation of models as diagrams in accordance with the underlying modeling language notation. The trend continues today with the new modeling languages and CASE-tools emerging to facilitate describing different views of the system as *viewpoints* of stakeholders. The IEEE Standard 1471-2000 [Hil00] describes viewpoints as a mean to establish the languages or notations enabling reusable, domain-specific architecture description standards [MEH01].

Object-oriented analysis and design is a complex task requiring viewpoints from requirements analysis, design, and modeling. This raises the need for stakeholders to work together and collaborate on models. Collaborative modeling refers to the process where a number of people actively contribute to the creation of a model [Rit09]. Renger et al. [RKdV08] describe collaborative modeling as; *The joint creation of a shared graphical representation of a system*. Collaborative modeling benefits by encouraging participants to work together, ask questions, explain, and justify opinions [RS05] [Sol01] [WTF95]. A major criterion for any effective collaborative session is forming the right group composition [DVDB03]. Frederiks et al. [FVdW06] highlights two main roles associated with collaborative modeling, namely *domain expert* who provide input to the *modeling expert* who is responsible for creating the formal model based on the input.

During the early stages of requirements engineering domain expert and the modeling expert is involved in brainstorming sessions to develop a common understanding of the system under development. This process accompanies elements of analysis of the problem domain as well as the solution domain. Domain experts provide critical input to the modelers to help them understand and articulate the abstractions of the application domain and to explore design alternatives. Since domain experts are usually not proficient with the modeling tools and techniques used by the software engineers, their participation is restricted to the role of knowledge source and have no direct input in creation or transformation of the model. On the other hand, modelers with limited or no awareness of the domain are forced to make sense of the information provided by the domain experts, leading to miss-communication, information loss, and rework.

Active participation of the domain experts is constrained by the very basic fact for whom the modeling CASE-tools are designed and the collaboration style supported. Traditional modeling CASE-tools are primarily designed considering analysts and modelers as the prospective users with the goal to allow creating precise, archival designs as formal models. Recent work employing technological enablers such as multi-touch surfaces [BM14] have tried to address the need of face-to-face or collocated collaboration among modeling participants. Still, the need to encourage and enable domain experts participation facilitated by modeling tools in collaborative modeling session remains a challenge and is the main topic of investigation in this dissertation.

This dissertation aims at addressing the limitations of existing modeling tools to support the collaboration between domain experts and modelers by identifying and evaluating new and intuitive interfaces with the focus on improved usability. The ISO 9241-11 standard defines usability as; *extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use* [ISO98].

1.1 Problem

Chervany et al. [CL98], and Albizuri et al. [AR00] highlights usability issues of existing modeling CASE-tools. Lahtinen et al. [LP03] specify that bugs, poor design, and inconsistencies are common issues attributing to the lowered usability of modeling CASE-tools. The current generation of modeling CASE-tools is based on the similar interface style of graphical node editors where a user can drag-&-drop items from the palette. These tools resemble an ordinary drawing tool or pen and paper with a canvas to draw models visually, with important operations hidden in the user interface making them hard to use for infrequent and non-expert users.

Due to the complexity and overhead involved in using traditional modeling CASE-tools, frequently individuals tend to organize information either using the pen, paper, and whiteboards, or any other idiosyncratic mediums during early requirements elicitation phases. Since information is captured sporadically, an additional overhead of merging different models representing views of every stakeholder into a single common understanding as a design is introduced implicitly. Subrahmanian et al. [SKL$^+$93] highlight that the practice of design involving multiple representations, disciplines, and stakeholders introduces the need to broaden the horizon of modeling techniques.

Understanding modeling concepts and the language notations is frequently not sufficient for users to get started and become productive with a modeling tool, as they are also required to learn tool-specific usage patterns. For example, the same use case can have variances in the execution steps from one modeling CASE-tool to another, which causes even the most experienced modelers to make errors and spend time rectifying them. Such issues can be attributed to the fact that the underlying modeling language describes the semantics of the language but do not provide a specification for the tool implementation. On the one hand, this gives freedom to the implementer of the modeling notation to develop the tool without any restrictions, but this also means the same modeling use case can be implemented differently by different CASE-tool.

While basic tool usage knowledge can be acquired rather quickly for simple modeling tasks, *model transformation* can be challenging even for the expert modelers due to the unfamiliarity with the model or modeled phenomena, or complex model with a large number of model elements, if accompanied with unfamiliarity to the tool. In this dissertation we employ the term *model transformation* to refer to the most common transformation techniques of model refactoring and model refinement. *Model refactoring* is a transformation process to improve the structure of a model while preserving its behavior [ZLG05]. *Model refinement* is a transformation that adds more detail to an existing model [VDSJM07]. As oppose to traditional refactoring and refinement techniques with the focus on the source code, this dissertation refers to the design models at higher levels of abstraction for refactoring and refinement.

Usability of modeling CASE-tools can be enhanced, e.g. by adding natural, and intuitive tool interfaces. Human-Computer Interaction (HCI) researchers have provided substantial evidence that multimodal interfaces can expand the accessibility of interfaces to diverse and nonspecialist users [OC00]. Jaimes et al. JS07 define a multimodal interface as; *a multimodal system is simply one that responds to inputs in more than one modality or communication channel.* These interfaces encompass both parallel and sequential usage of different modalities such as speech, touch, gestures, body movements in a coordinated manner [Tur14][Ovi99].

Mignot et al. [MVC93] studied the use of speech and gesture and found that gestures are effective for simple and direct commands, while speech is more effective with the abstract commands. Speech is an effective interaction modality for novice and occasional users. Further, Oviatt et al. [OCL04] observed that users spontaneously respond to increase in their cognitive load by shifting to multimodal interactions. Cognitive load refers to the amount of information that can be held in short-term memory and is attributed to the mental effort invested for a certain task [Swe88].

Recent work employing multi-touch tabletops and interactive whiteboard enable new ways of interaction in collaborative environments [CGH03][FHD09]. Interactive whiteboards are touch-sensitive boards that allow controlling a computer connected to a projector. Existing research demonstrates that interactive whiteboard foster interactivity, participation, and collaboration among users [SHWM05]. Wu et al. [WG05] describe RoomPlanner, a prototype application for room furniture layout designing which allows users to collaborate on a multi-touch tabletop using touch gestures. Touch gestures reduce the number of primitive touch events required to perform a complex operation by capturing additional information in the user interaction. Kurtenbach et al. [KH90] distinguish between touch and touch gesture as;

> A touch gesture is a motion of fingers that contain information, and from a human point of view have significance. A simple touch, on the other hand, is not a gesture as the motion of a finger to touch a spot on the interface is neither observed nor significant.

Usability issues in the current crop of modeling tools, and the need to encourage the participation of domain experts leads us to the following **problem statement**, which is addressed in this dissertation:

> We can enhance the usability of modeling tools by employing multimodal interfaces such that it improves the efficiency of the modeler, and reduces the learning curve required to be productive. This improved usability will encourage the participation of domain experts in collaborative modeling sessions.

Based on the problem statement this dissertation investigates the following hypotheses:

> **H1:** A multimodal interface utilizing an interactive whiteboard and speech input improves the *efficiency* of modelers during modeling and model transformation activities.

Acceptance of the hypothesis (H1) that multimodal interface improves the *efficiency* of modelers leads us to the investigation of the follow up hypothesis:

> **H2:** Modelers find multimodal interface employing touch, speech and touch gesture input modalities more *useful* and *practical* as opposed to an interface using only touch and speech input modalities.

A challenge in designing multimodal interface is to identify, evaluate, and associate interaction modalities to the system model. Technology products such as the Leap Motion controller [1], Microsoft Kinect [2], or Ideum's touch display [3] provide possibilities for multiple users to interact with the system simultaneously [Seb09]. This concurrent environment raises the need of a formal process for identifying and evaluating the combination of modalities with a focus on improved usability in each ***use case***.

A use case describes the behavior of the system from an ***actors*** point of view. It captures function provided by the system as a set of events that yield a visible result for the actors [JCJO92]. Use cases can be described in different levels of granularity. For example, an essential use case is a simplified form of use case, an abstract scenario for a complete and intrinsically useful interaction with a system from the perspective of the user [Con95]. While a generic use case allows capturing the flow of events between user and system in the form of user step and system step, but with no focus on interaction details such as whether the interaction is unimodal or multimodal in nature. To best of our knowledge, even user interface design techniques provide no means to model multimodal interactions.

[1]https://www.leapmotion.com/product/vr

[2]http://www.xbox.com/en-US/xbox-one/accessories/kinect

[3]http://ideum.com/touch-tables/platform/

This information if available can allow the system designers to evaluate each use case execution over the usability characteristics identified by Nielson [Nie92] and further benefit in identifying alternate steps (execution paths) with new interaction possibilities aiming at reduced learnability and improved efficiency. Capturing interaction details additionally allow recording the rationale for the selection of a unimodal or multimodal interaction in a certain usage context (for example, considering touch input over speech in an outdoor environment).

1.2 Research Approach

With the goal of improving the usability of modeling CASE-tools by employing intuitive multimodal interfaces for early stage requirements engineering process, this dissertation employed a three steps approach. *Understanding the problem domain*, *devise tools and frameworks*, and *validate the research hypotheses*. The first step is realized with a strong emphasis on understanding the issues affecting the usability of modeling CASE-tools, and how the multimodal interfaces could be employed as a mean of improving modeling, and model transformation process. The second step devises the M^{3}framework with a unified model for capturing multimodal interaction information and associating it with the use case model during the design process of new multimodal interfaces. The M^{3}framework is employed in a bootstrapping process during the development of MiNT framework. MiNT framework was designed and developed to allow modeling tool developers to prototype modeling CASE-tools with multimodal interfaces.

In the third step, two controlled experiments are performed to evaluate the hypothesis of this dissertation. MiNT Eclipse, a reference implementation of MiNT framework was developed to evaluate the first hypothesis (H1). Results from the first controlled experiment provided evidence that multimodal interface improves the efficiency of modelers. Based on the observation made during the first experiment, as well as expert feedback, and literature review we developed a second reference implementation of the MiNT framework namely; MiNT Mobile to evaluate the second hypothesis (H2). During the development of reference implementations, we continuously sought and incorporated feedback from expert modelers in pilot studies for identification of speech commands and touch gestures for modeling and model transformation.

Since the usability of modeling tools is the main concern of this dissertation, we employed multimethod research approach for the evaluation of the hypothesis [BH89]. More specifically mixed methods research was used that combines elements of qualitative and quantitative research approaches [JOT07]. The controlled experiment conducted to evaluate both hypotheses rely on a set of inquisitive [SSL08] techniques for collecting qualitative and quantitative data. Shadowing and observation were used as a mean to understand how modelers work with unimodal and multimodal interfaces to perform modeling and model transformation tasks. An interview questionnaire format was employed to capture the subjective evaluation of the multimodal interfaces by the modelers.

1.3 Outline of the Dissertation

This dissertation is structured as follows:

Chapter 1 introduces the usability issues with modeling CASE-tools and their limitations for early stage requirements engineering process. Multimodal interfaces are identified as a mean to address the usability problems of modeling interfaces. Further, the research approach is described.

Chapter 2 presents the general foundations of this dissertation with the emphasis on modeling and model transformation, collaborative modeling, multimodal user interface, and usability engineering.

Chapter 3 introduces the M^3 framework with a meta-model that allows capturing modalities information of the interactions and associating it with the use case.

Chapter 4 describes the elements of multimodal modeling and model refactoring, and provide a basis for developing multimodal modeling interfaces.

Chapter 5 presents MiNT framework aimed at assisting multimodal interface developers for rapid prototyping. Afterward, two reference implementation of the framework is described, namely MiNT Eclipse and MiNT Mobile.

Chapter 6 describes a controlled experiment conducted to investigate the applicability of the multimodal interface employing interactive whiteboard and speech as an input modality for modeling and model refactoring.

Chapter 7 describes the second controlled experiment conducted to evaluate the usefulness and practicality of two different multimodal interfaces for modeling and model refactoring.

Chapter 8 concludes the dissertation and discusses the directions for future work.

Chapter 2

Foundations

Models allow understanding complex information and navigating around designs. This is especially true for *Design Space Exploration* (DSE), which facilitates identifying and evaluating design alternatives [KJS11]. Some common scenarios for DSE are:

1. *Prototyping:* Create and evaluate the impact of design decisions before implementation.
2. *System design:* Create and evaluate complex system designs in greenfield engineering projects. Complex software systems consist of a large number of components, which can be identified and evaluated over design goals.

DSE allows the identification of design alternatives in the design process as well as the software maintenance scenarios.

In section 2.1 we describe modeling and the state-of-the-art modeling CASE-tools. Section 2.1.1 describes the model transformation process that aims at improving the structure of existing model along with facilitating behavioral changes by adding new information. Section 2.2 focuses on collaborative modeling and highlights the importance of involving application domain experts in the collaborative modeling sessions. In section 2.3 we describe multimodal interfaces and the role of multimodal interaction for natural human interaction. Finally, section 2.4 focuses on the *usability* of multimodal interfaces from the viewpoint of software engineering.

2.1 Modeling in Software Engineering

Software Engineering is a problem-solving domain in which models are used to visualize and understand a system that may or may not exist at the time the model is created. A model is an abstract representation of a problem with the focus on the relevant aspects and ignoring all other [BD10]. Models are created to serve particular purposes, for example, to present a human-understandable description of a system for communication or to capture design information that could be transformed into a different model [FR07].

The most common approach to modeling is based on diagram sketching, in which the modeler or the designer freely creates diagrams on paper or using a paint program. These sketches are informal models, and they do not use any formal language notation. Once the common understanding of the informal model has been established among the stakeholders, the diagram is digitized using a formal modeling notation. Digitization of sketches and transferring them to a formal notation has been researched by several researchers since sketching is a natural part of human problem-solving. Hammond et al. [HD06] demonstrated that informal models created using sketches could be translated into formal models. They recognize a set of objects by their geometrical properties from sketches. Plimmer et al. [PF07] introduced a sketch tool framework with the objective to allow quick prototyping of domain-specific sketching tools. Damm et al. [DHT00] investigated shape gestures in the sketch drawn on an electronic whiteboard. Their approach aimed at facilitating collaboration between modelers using informal and formal elements.

The second common approach for modeling is to employ tools with pre-defined graphical elements. These tools focus on producing implementation and deployment artifacts from models under the umbrella of Model Driven Development (MDD) [FR07]. Example of such CASE-tools are Visual Paradigm [1], Enterprise Architect [2], and Eclipse Papyrus [3]. Mellor et al. [MCF03] describe MDD as *Model-driven development is simply the notion that we can construct a model of a system that we can then transform into the real thing*. Since the objective of these tools is to support the complete software life-cycle model, they are complex to use and requires expertise to be productive.

Researcher such as Mackay et al. [MNB03] distinguish between light-weight and heavy-weight tools by the amount of functionality they provide. Heavyweight tools offer a much functionality that relates to the problem domain, whereas lightweight tools provide only

[1]https://www.visual-paradigm.com/

[2]www.sparxsystems.de/

[3]https://eclipse.org/papyrus/

essential functionality that is necessary to support a certain stage of software life cycle model. Learning and using the heavy-weight tools can be overwhelming because of a large number of inbuilt features. Lightweight tools are more suited towards a particular activity (for example focus only on analysis phase) without cluttering the user interface with unwanted and unused features. Biddle et al. [BNT02] present a list of light-weight CASE-tools for different phases of software lifecycle. A web-based CASE tool for creating UML sequence diagram is presented by Khaled at al. [KMB+02].

General purpose diagramming tools such as painting applications, OmniGaffle [4], SmartDraw [5] or Microsoft PowerPoint [6] does not follow the semantics of any modeling language. In this dissertation, we do not consider diagramming tools for modeling.

Models undergo transformation either to improve the existing solution or to extend the solution with new features. In the next section, we describe model transformation process an essential part of the modeling process.

2.1.1 Model Transformation

Model transformation is the generation of a target model from a source model, following a transformation definition. A ***transformation definition*** is a set of transformation rules that describe how a source model can be transformed into a target model [KWBE03]. The input to any model transformation process is a source model with existing model elements, and a transformation requirements describing the changes to be done to the model. The process itself consists of the following steps:

1. Identify the model elements in the source model that requires transformation and modify these elements following the problem description.
2. Improve the structure of the model while preserving the existing behavior.
3. Introduce new model elements to add the behavior as described in the problem description.

Step 1, Step 2, and Step 3 are executed iteratively until the target transformation is achieved.

Step 2 is also known as model refactoring. ***Model refactoring*** is a process used to improve the structure of a model while preserving its behavior. Step 3, also known as ***Model refinement***,

[4]https://www.omnigroup.com
[5]https://www.smartdraw.com
[6]https://office.live.com/start/PowerPoint.aspx

is a process that adds more detail to an existing model[VDSJM07]. Model refinement is frequently performed along with model refactoring to improve the design or adapt to new requirements during model evolution. It can invoke creation of new model elements, update or remove existing ones with the aim of reorganization or adding detail to the contained information [SK03]. Additionally, based on the transformation requirements design patterns are employed to improve the existing model [FCSK03].

In the next section, we introduce the notion of *collaborative modeling*, an important aspect of the modeling process. It provides the foundation of designing modeling interfaces for collaboration among stakeholders.

2.2 Collaborative Modeling

Collaborative modeling refers to the process where a number of people actively contribute to the creation of a model [Rit09]. Renger et al. [RKdV08] describe collaborative modeling as; *The joint creation of a shared graphical representation of a system.* Collaborative modeling benefits by encouraging participants to work together, ask questions, explain, and justify opinions [Sol01]. As software engineering becomes an increasingly complex and heterogeneous discipline, it raises the need for collaboration among stakeholders. The value of collaboration has long been identified in the Computer Supported Collaborative Work (CSCW) research [Gre89].

Johansen [Joh88] further established the needs of collaborative teams into four basic categories: same place (colocated), different place (remote), same time (synchronous), and a different time (asynchronous). A majority of the current generation of modeling CASE-tools support remote and asynchronous style of collaboration by utilizing model repositories. Model repositories are storage systems for models that are mostly focused on persistence and concurrent access over a distributed infrastructure and allows [KRM+13]. Another group of modeling CASE-tools aims at satisfying the need of synchronous modeling by using a single, shared instance of the model which is edited by multiple users at the same time [Pin03]. While only recently researchers have started to study the needs and importance of colocated and face-to-face style of collaboration. For example, Wu et al. [WG05] describe the importance of collaboration and communication in software design, and motivate the need to support these activities in software design tools. They introduce Software Design Board a modeling tool that allows modelers to work on their computer desktop in single user mode, as well as employing electronic surface in a collaborative session. The availability of

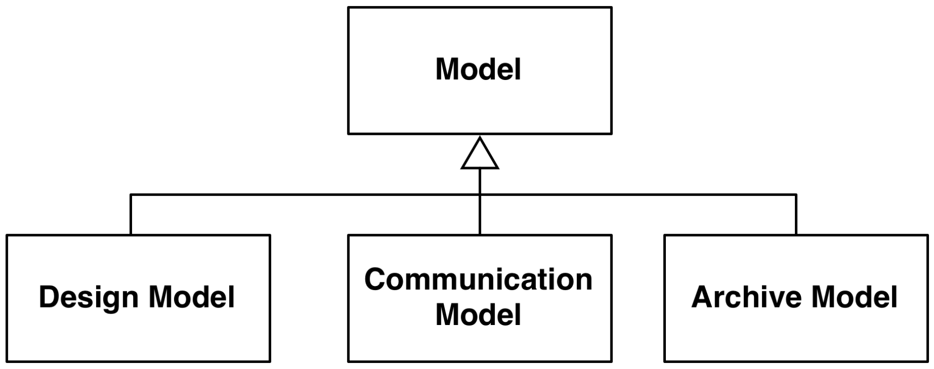


Fig. 2.1 Traditional classification of models

multi-touch tabletops such as Platform [7] and TableConnect [8] are enabling new collaboration interfaces [BB12], [A^{+}13].

In the past models were classified in three categories (see Figure 2.1) based on the activities they support:

> **Design models** provide a representation of the problem that enables stakeholders to understand and reason about the solution. In software engineering design models were specifically used as a formal representation of the system that can be understood by CASE-Tool. Design models were presented in a language with formal semantics (such as UML or SysML) and could be transformed into a representation understood by a computer. For that reason, design models were required to be correct, complete, consistent, and unambiguous.
>
> **Communication models** serve stakeholders to establish a common understanding of the problem to be solved. In software engineering communication models include a broad range such as a model sketched on paper or a whiteboard to capture designs informally. Since the primary purpose of a communication model is to establish a common understanding among the modeling participants as opposed to providing a basis for a specification they can be incorrect, incomplete, inconsistent, and ambiguous in nature.
>
> **Archive model** provide a compact representation for storing the design and its rationale for future reference. Design rationale is “the historical record of the analysis that led to the choice of the particular artifact or the feature in question” [LL91] and allows stakeholders to become familiar with early decisions.

[7] http://ideum.com/touch-tables/platform/
[8] http://www.fingermarks.de/

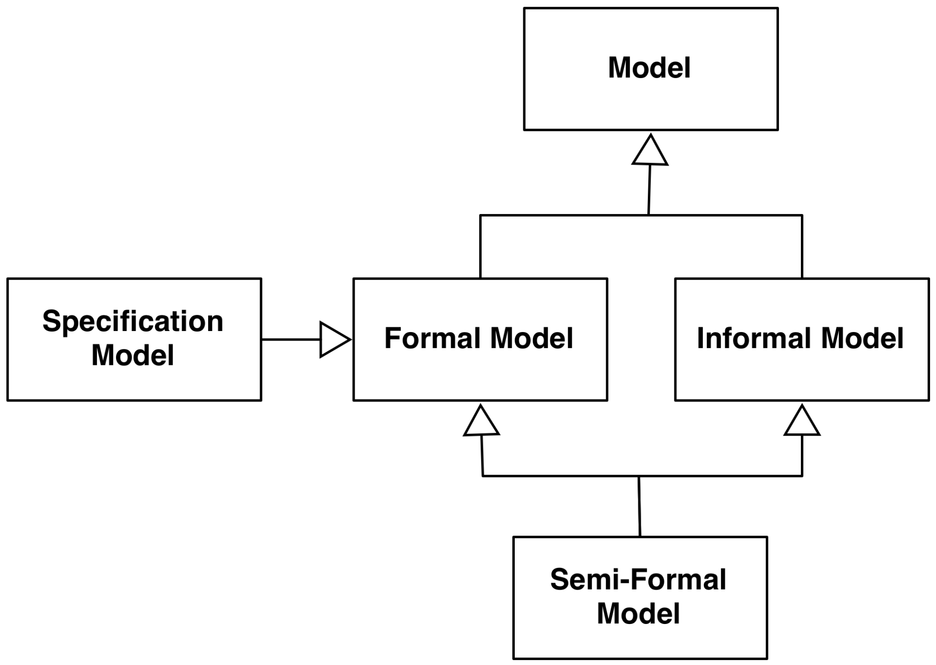


Fig. 2.2 Semi-formal model for multimodal interaction

Traditionally, the collaborators of a communication model were application domain experts and solution domain experts. The collaborators of in a design model collaborators were solution domain experts such as designers and developers. Application domain experts with their limited or no CASE-tool familiarity could only collaborate on design models using formal textual annotations or change requests that had to be executed by a CASE-tool expert. Working with design models required knowledge of either front-end CASE-tools for requirements, specification, planning, and design or integrated CASE-tools supporting the complete software life-cycle from analysis to maintenance. On the other hand, communication models were usually created with tools such as a pen or with generic drawing programs.

In 2008 Renger et al. [RKdV08] already highlighted the importance of active involvement of domain experts in the modeling process to improve the feeling of ownership, acceptance of the model and the decisions derived from it. Recent advances in continuous integration and continuous delivery enable frequent iterations where domain experts should provide their feedback on the models. The emergence of continuous software engineering as a new discipline emphasizes the importance of continuous involvement of domain experts even more because now even informal models such as mockups can be used to generate executable prototypes.

In the continuous software development paradigm, the traditional distinction between design models and communication models is therefore no longer valid and has become blurry. In this

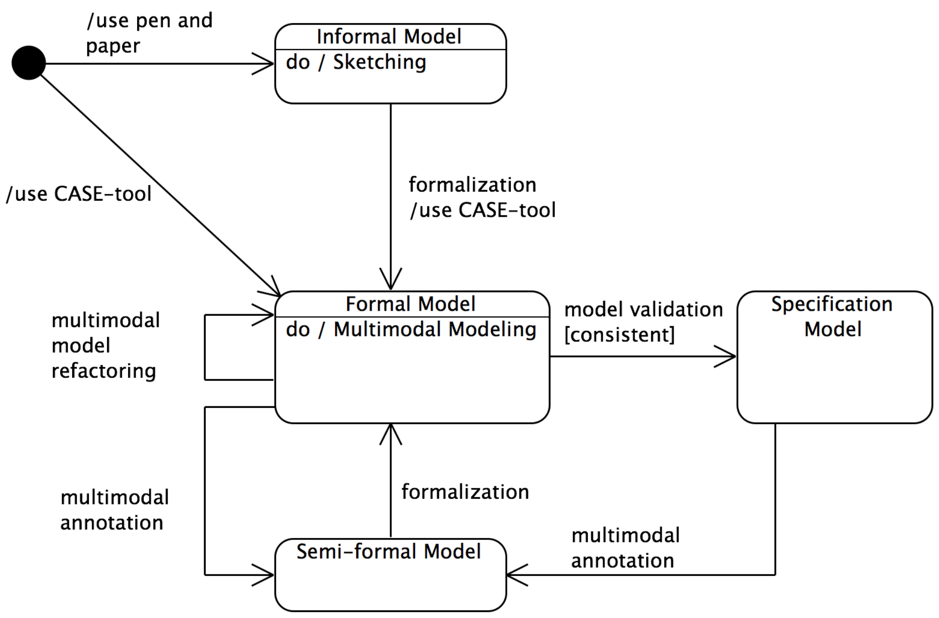


Fig. 2.3 State diagram showing the transition betwen different states of a model

dissertation we introduce the notion of *semi-formal model* (see Figure 2.2). A *semi-formal model* inherits all the property of a formal model as well as all the properties of an informal model. A formal model can further be classified as a specification model which is consistent with the requirements of the system.

Figure 2.3 shows the transition to different states of a model during multimodal modeling. A modeler starts with creating an informal model by sketching using a pen and paper. Once a common understanding is established the informal model undergoes a formalization process using a modeling CASE-tool with a multimodal interface. The modeling CASE-tool adheres to a language with formal semantics such as UML or SysML. Modelers can also directly start with creating a formal model using the multimodal modeling. Models undergo transformation using multimodal model refactoring to create a target model. A model which is consistent and can serve as a basis for code generation is derived using model validation process as specification model. During the design review meetings and collaborative modeling sessions application domain experts and other stakeholders employ multimodal annotations in the form of textual notes, audio notes or hand draw shapes to create a *semi-formal model*. A semi-formal model can then undergo formalization process to create a formal model, which includes the changes requested as part of multimodal annotations.

This dissertation aims at improving the collaboration on software engineering design models by including non-CASE-tool experts such as application domain experts and allowing them to interact with models naturally. Application domain experts, as well as solution domain experts, can collaborate on models naturally using multimodal interactions using speech and gestures input modalities. We assume that natural multimodal interactions will further reduce the complexity of performing model transformation such that even application domain experts can easily perform design changes which were traditionally restricted to expert modelers.

2.3 Multimodal User Interface

In human-human communication, different modalities corresponding to human senses play an important role. The human senses are sight, touch, hearing, smell, and taste. Almost any natural communication among human involves multiple modalities [SPH98].

Multimodal interfaces aim at harnessing the natural form of human communication for multimodal human-computer interaction. Multimodal interaction in the domain of human-computer interaction is a class of interaction in which a human-computer interface employs more than one modality (enabled by one or more devices) for multimodal input and multi-modal output interactions. Bolt [Bol80] used speech and gesture to allow the user command simple shapes on a large-screen graphics display surface. Oviat et al. [Ovi03] describes multimodal interface as;

> Multimodal interfaces process two or more combined user input modes such as speech, pen, touch, manual gestures, gaze, and lip movements in a coordinated manner with multimedia system output.

Multimodal interfaces process two or more combined input modes such as speech, pen, touch, manual gestures, gaze, and lip movements for multimodal input. The multimodal output is produced by using two or more output modes such as visual display, audio, and tactile feedback is combined to present the state of the system to the user. By employing multiple modalities, a multimodal interface provides the user freedom in selection of the modality to interact with the system [OC00]. For example, hearing impaired user will prefer speech as an input modality, and the visual display or tactile feedback for the multimodal output. The multimodal interface provides the feasibility for interchanging the modalities based on their suitability for a particular usage context and environment. For example, a user will prefer to use gesture or keyboard input in a noisy environment as oppose to employing speech for interaction.

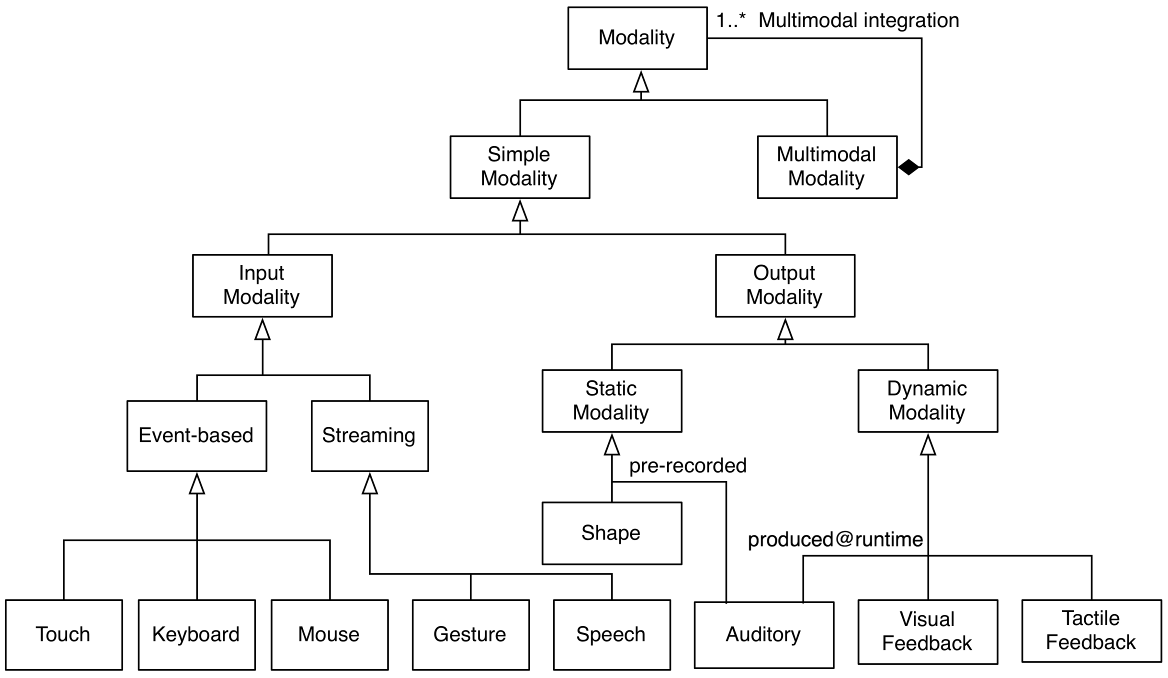


Fig. 2.4 Input and output modality. Adapted from [OS04]

In multimodal interaction, two or more primitive interaction are employed to perform the action, for example pointing at an object and speaking a voice command. Modality can either be of simple or composite (see Figure 2.4). A simple modality represents a primitive interaction, whereas a composite modality integrates other modalities to enable multimodal interaction. Primitive interactions employ only one modality at a time (unimodal in nature) for interaction. Modalities can be classified as input modality and output modality. Section 3.1.1 describes input and output modality as part of the unified meta-model of the M^3 framework.

Multimodal integration is fundamental towards integrating multiple modalities to create natural interfaces with multimodal input and multimodal output capabilities. Multimodal integration consists of multimodal fusion and multimodal fission. Multimodal fusion (see Figure 2.5) plays a crucial role in combining and interpreting various input modalities for input whose meaning can vary according to the context, task, user and time [LNR$^+$09]. On the other hand, multimodal fission (see Figure 2.5) distributes the output over multiple channels corresponding to human senses [Wah03]. Dumas et al. [DLO09] divide multimodal human to computer interaction into four different states. First decision state in which the communication content is prepared, second action state where means of communication are selected, third perception state where multimodal interfaces receive the message through hardware enablers or sensors, finally fourth the interpretation state where the multimodal

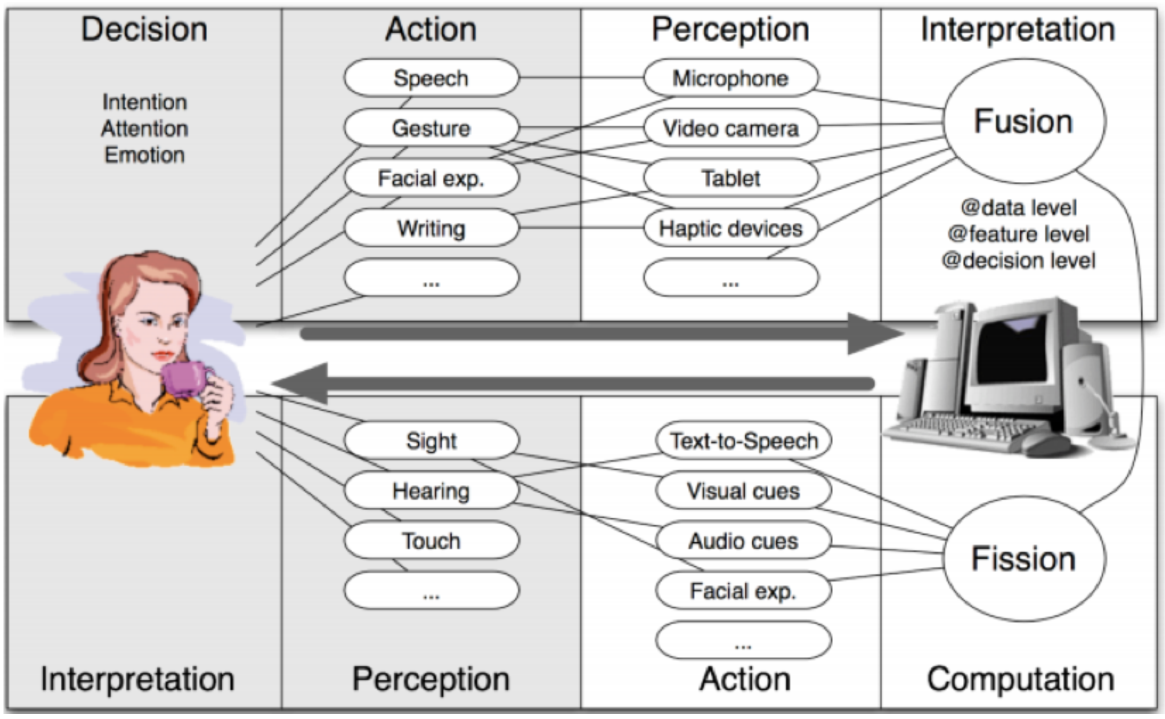


Fig. 2.5 A representation of multimodal man machine interaction loop from [DLO09]

fusion occurs to derive information from the interaction. Further, in the computational state following the interpretation of human interaction, a response is generated and transmitted in the action state by using multimodal fission.

Multimodal interfaces seek to utilize the natural human capabilities to communicate via speech, touch, gesture, facial expression, eye movements, and other modalities since human beings naturally interact using multimodal interaction. Multimodal interface that employs pen and speech has shown to improve the efficiency of visual-spatial tasks by 10% [Ovi97]. They observed that multimodal interfaces offer improved error handling and reliability as users made 36% fewer task-critical content errors with a multimodal interface than with a unimodal interface. Similarly, Pausch et al. [PL91] showed that adding speech to a drawing application reduced time to completion by up to 56%, with results showing an average reduction of more than 21%.

This dissertation employs multimodal interaction for collaborative modeling. The hypothesis is that even domain experts who are not familiar with CASE tools can participate in collaborative modeling activities, making modeling more natural again. Multimodal interfaces aim at making human-computer interaction natural by improving the usability. In the following section, we describe the characteristics of usability.

2.4 Usability Engineering

The success of any software system depends on several different factors such as functionality, performance, reliability, maintenance, and usability [May99]. Usability is defined in the ISO 9241-11 standard as *extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use'* [ISO98].

Neilsen identifies five major characteristics of usability.

- *Learnability* describes how easily and intuitively a user can get started with the product to achieve his/her use case.
- *Efficiency* describes how fast users can operate a system once they are through the learning curve.
- *Memorability* defines how much effort is required for a returning user to get productive with the product.
- *Error handling* describes how the product reacts to the user errors, and mechanisms to provide informative feedback and exit strategy.
- *Satisfaction* describes the overall user experience.

In the design and development of interfaces with multimodal input and multimodal output capabilities, usability plays an important role. Multimodal interfaces employing speech and gesture can reduce the number of primitive user interactions and thus subsequently improve the usability and user-experience. We believe that multimodal interfaces can further help in overcoming the gulfs as identified by Norman in his work [Nor86]. The two gulfs, which need to improve human usage of the systems, are the gulf of execution and evaluation. The gulf of execution represents the effort that the user has to make to translate their goals into the action sequences, which when applied to the system, will achieve the goal. The gulf of evaluation represents the effort the user has to make to understand the state of the system as a result of their actions. The natural form of multimodal interaction can reduce or eliminate the need to learn tool specific usage. Similarly, system state conveyed over multiple channels (multimodal output) are more effective in seeking users attention to communicating the system state.

Coutaz and colleagues [CNS$^+$95] define properties to assess the usability of multimodal input and multimodal output interaction. Vernier et al. [VN00] in their work describe a framework for combining and characterizing output modalities for classifying existing output systems and evaluating the usability of an interface.

To allow developers of multimodal interfaces capture multimodal interaction during the design process of new multimodal interfaces, we developed M^3 framework, which is described in the following chapter.

Chapter 3

Multimodal modeling (M^3) Framework

Capturing interaction information at design-time allows designer to answer two important questions;

1. How does a user interacts with the tools interface using multimodal interaction?
2. How does the system responds to the user using multimodal output?

In this chapter we present M^3 framework allows to capture the different aspects of multimodal interaction and associate it with the use case model from software engineering. The M^3 framework allows designers to capture human-computer interaction and computer-human interaction information during the design process on new multimodal interfaces.

Section 3.1 describes the M^3 framework with the unified meta-model. In section 3.2 we present *diagram presentation modality* model instantiated using the unified meta-model of the framework. The model provides evidence for the applicability of the proposed framework in investigating multimodal integration. Section 3.3 describes the application of M^3 framework during the design and development of a multimodal modeling tool.

3.1 M^3framework

The M^3 framework combines the generic multi-modality model with the use case meta-model. The generic multi-modality model is adopted from the work of Obrenovic et al. [OS04]. The presented models give a high-level view of the various types of interaction modalities, the relationship between them, and also a clear description of input and output modality, with the computer as a viewpoint. A challenge in designing multimodal interface is to identify, evaluate, and associate interaction modalities to the system model. While a use case allows capturing the flow of events between the user and system in the form of **UserStep** and SystemStep, it does not focus on interaction details such as whether the interaction is unimodal or multimodal in nature. To best of our knowledge, even user interface design techniques provide no means to model multimodal interactions. The unified meta-model of M^3 framework aims at overcoming these challenges by allowing capture multimodal input and multimodal output information during the design phase of a multimodal interface.

3.1.1 Generic Multi-modality Model

In the domain of human-computer interaction, modality is defined as a mode of communication according to human senses and computer input devices activated by humans [JS07].

The modalities can be expressed with a composite pattern as shown in Figure 3.1. A modality can either be simple or multimodal [1]. Simple input or output modalities are common in system interfaces (e.g. An electronic calculator with key input and textual output on liquid crystal display (LCD). Multimodal integration is used for multimodal modality. For example, an interface with multimodal input allows the user to combine a touch gesture with speech input. Similarly, a system with multimodal output capabilities employs multiple output modes. For example, when clicking a button in the user interface the system can visually respond with a picture of a clicked button and a clicking sound. Existing research work describe the challenges involved in the design of multimodal interfaces [Seb09] [Ovi99].

Event-based modalities and streaming modalities are a form of input modality. An event-based modality takes input in the form of discrete events such as touch, keyboard input or mouse input (see Figure3.2). An input event occurs when a user presses a key on the keyboard, uses the touch input on touch-sensitive hardware or uses the mouse to click on a certain user interface element. A streaming-modality is an input in the form of a continuous-time signal. This kind of input requires pre-processing by a system to decipher the action that the user wants to execute with the input. Examples of a streaming modality are gesture recognition and eye tracking. Another example of a streaming modality is natural language speech input that requires a speech recognition engine to use an acoustic model and grammar

[1] In [OS04] calls it composite modality

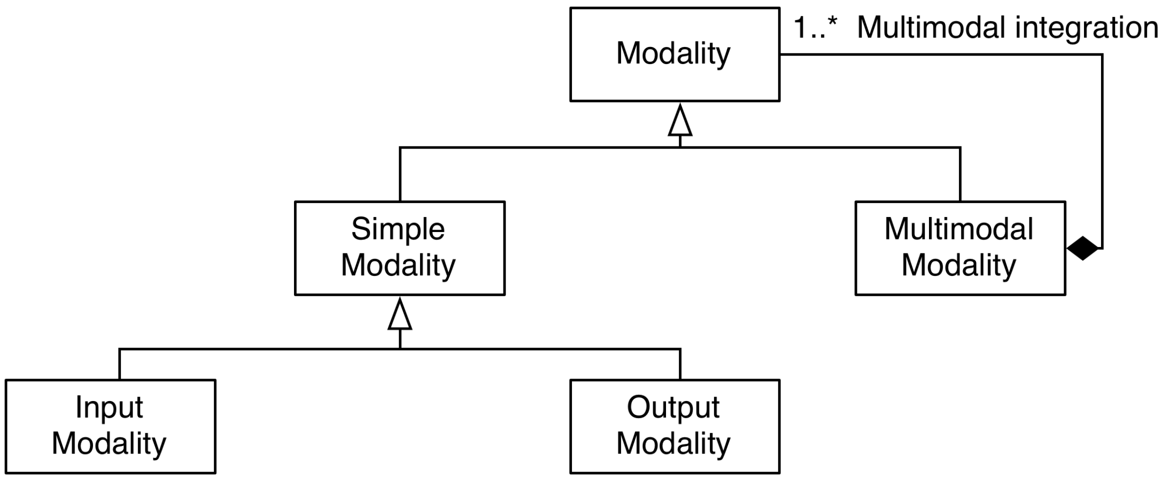


Fig. 3.1 Excerpt from the generic multi-modality model (UML Class diagram)

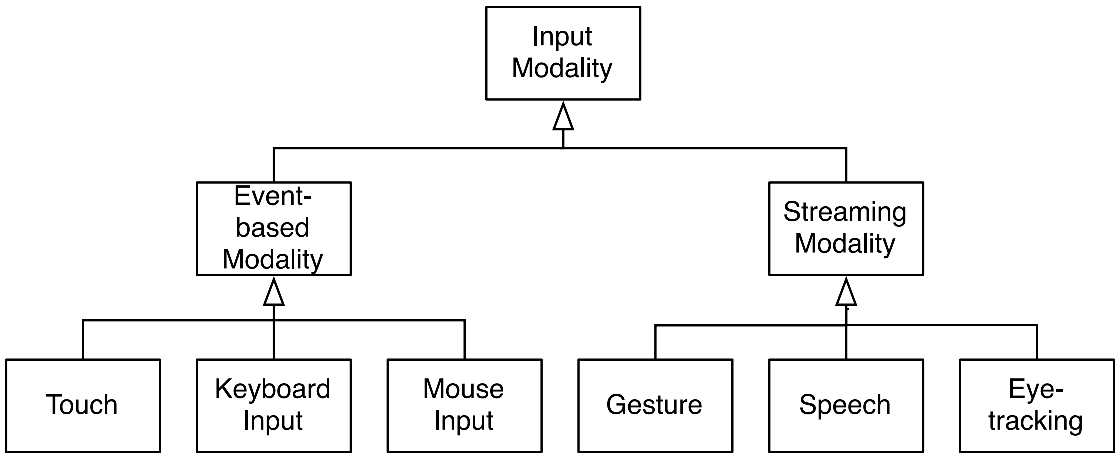


Fig. 3.2 Input modality model (UML Class diagram)

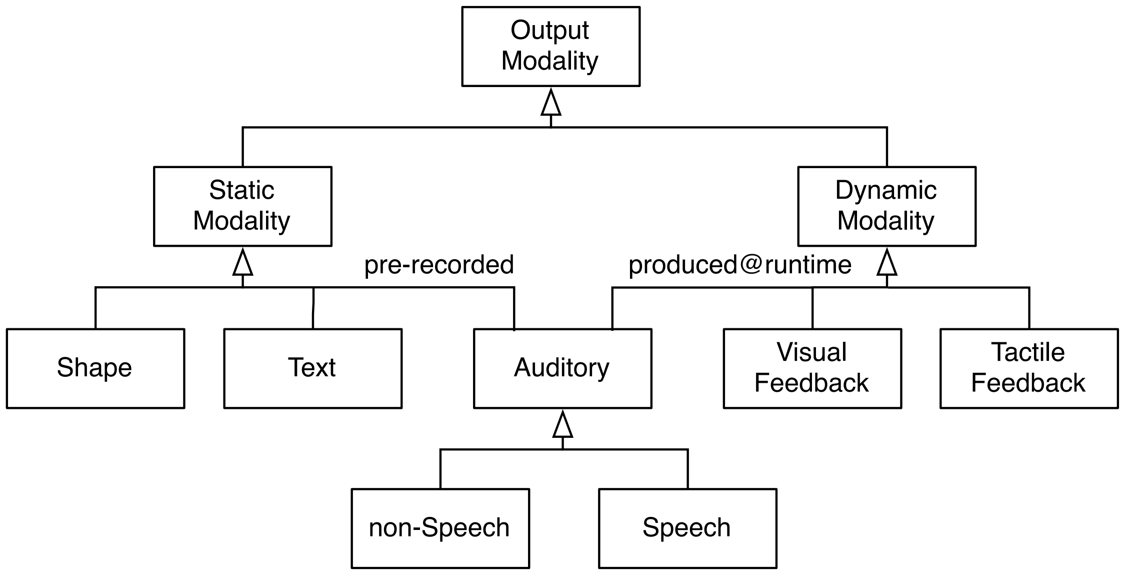


Fig. 3.3 Output modality model (UML Class diagram)

to translate from the speech into text. Once the text is available, the system can map it to a specific action.

Feedback is an important aspect of human-computer interaction. Renaud et al. [RC00] defines feedback as *the communication of the state of the system, either as a response to user actions, to inform the user about the conversation state of the system as a conversation participant, or as a result of some noteworthy event of which the user needs to be apprised*. In this dissertation, we use the term output modality synonymously with the term feedback. Output modalities can be categorized into two groups as static modality and dynamic modality (see Figure 3.3).

A static modality presents stationary data to a user. Examples of static modalities are shapes, text, icons, and sounds (called auditory in the class diagram). On the other hand, *dynamic modality* is sometimes produced by animating the static modality. For example, a static shape in a visual modeling tool when moved from one position to another presents a dynamic visual feedback to the user. An auditory feedback produced at runtime to convey the current state of the system is regarded as dynamic modality, whereas a pre-recorded auditory feedback is regarded static (see Figure 3.3).

In the following section, we present a meta-model for use cases which allows attaching these modalities to the interactions performed by the user and system in the event flow of a use case.

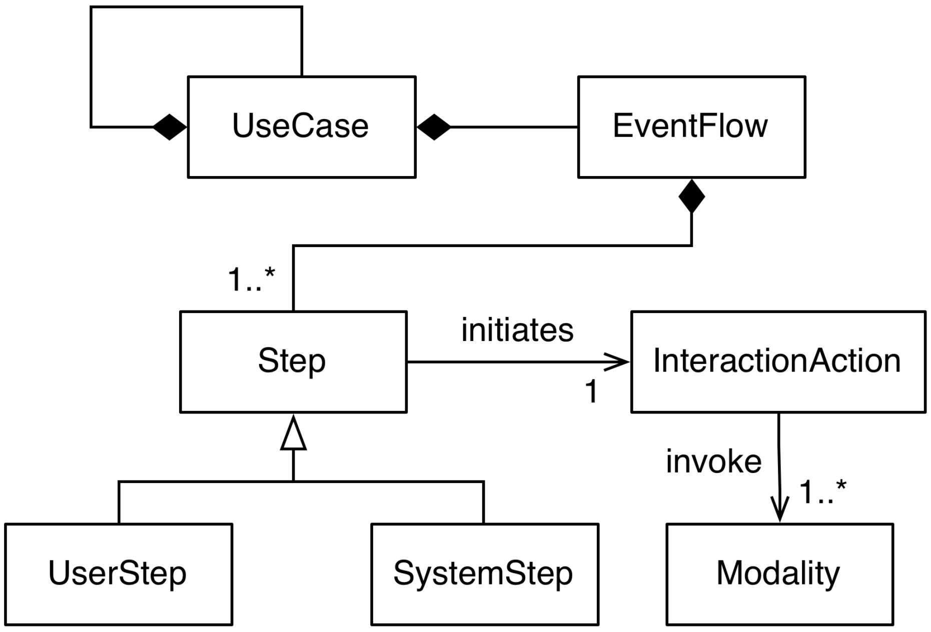


Fig. 3.4 Unified meta-model of M^3 framework (UML Class diagram)

3.1.2 Use case Meta-model

Jackbson [Jac93] defined a use case as a description of the behavior of the interaction between an actor and the system from an actor's point of view. Bruegge and Dutoit [BD10] extended this definition by describing the behavior with an **EventFlow**, which consists of one or more **Steps** each of them modeling an interaction. A **UserStep** captures the interaction of the actor with the system from the users point-of-view, while a **SystemStep** captures the system response. In M^3 each **Step** is associated with an **InteractionAction** class (see figure 3.4). The **InteractionAction** describes the interaction modality used during the **Step**. For example, when the actor touches the screen the interaction modality is of type **Touch** (see figure 3.2), when the system responds with a sound, the interaction modality is of type **Auditory** (see figure 3.3).

Step can initialize an **InteractionAction**, which in turn invokes one or more modalities made available from the generic multi-modality model (see Figure 3.1). For a *UserStep* input modality is employed for unimodal or multimodal input. Whereas for a **SystemStep** an output modality such as visual feedback or auditory feedback is employed for unimodal or multimodal output.

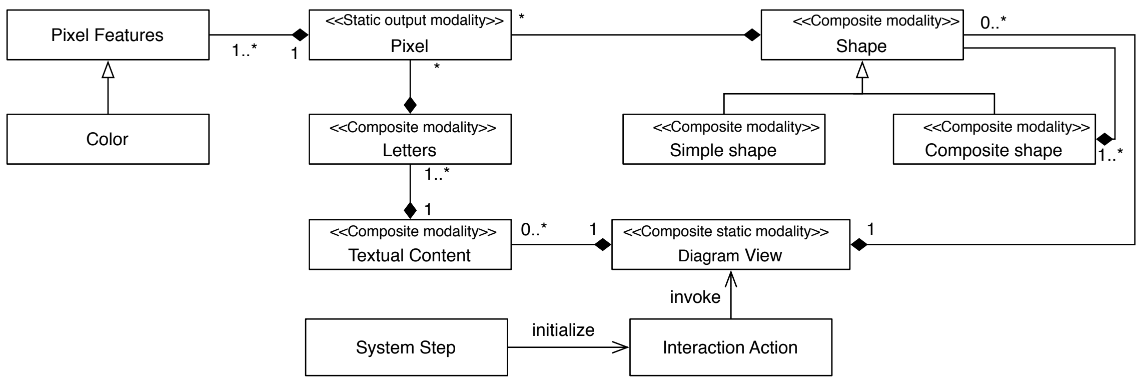


Fig. 3.5 Diagram presentation modality for the **open diagram** use case (UML Class diagram)

Step#	Step	**InteractionAction**	Modality
1	Open diagram (UserStep)	Point and click on the diagram in the tool	Mouse Input modality
2	Open class diagram (SystemStep)	Open diagram editor and show diagram	Diagram presentation modality (composite output modality)

Table 3.1 Event flow in the open diagram use case

3.2 Models for Visual Modeling Tools

This section describes the diagram presentation modality and diagram manipulation modality instantiated using the M^3 framework. Diagram presentation modality captures the static data shown to the user in a visual modeling tool. Diagram presentation modality is explained using generic use cases from UML diagramming tool for the ease of understanding.

3.2.1 Diagram Presentation Modality

The interface of visual modeling tool offers a diagram view, a palette to select new elements from and an additional set of tools to work with the diagrams. Diagram view is used to present existing models to the users and also serve as a workspace to create a new diagram and modify existing ones.

Diagram presentation modality (see Figure 3.5) is a complex modality that describes the static representation of a diagram as shown to the user in a modeling tool. At the core of the diagram presentation modality is *DiagramView*. A *DiagramView* consists of various output modalities as indicated in the figure. Based on the modeling notation of the tool, a model

Step#	Step	**InteractionAction**	Modality
1	Enable voice input (UserStep)	Tap on the voice command button	Touch Input modality
2	Digital volume unit meter (SystemStep)	Display digital volume unit meter	Visual feedback (static output modality)
3	Listening voice command (SystemStep)	Auditory feedback *listening voice command*	Auditory feedback (static output modality)
4	Create a class (UserStep)	Voice command *create class*	Speech input modality
5	Listening voice command over (SystemStep)	Auditory feedback *done listening voice command*	Auditory feedback (static output modality)
6	Class creation (SystemStep)	A new class is created on the canvas and is highlighted	Visual feedback (static output modality)

Table 3.2 Steps in create class use case use case

can have any number of output modalities such as images, videos, shapes, icons, and text. Shapes can be either simple or composite. Simple shapes represent the primitive shapes like lines and dots, while the composite shapes are made of simple shapes (for example, notation of a UML class with rectangles and text). All the elements in the *DiagramView* are complex output modality derived from the simple output modality *Pixel*. A *Pixel* can have several features such as color to provide distinct visual clues to a human user.

Table 3.1 presents the various steps involved in displaying diagram presentation modality for the **open diagram** use case. To initiate the use case user performs an **InteractionAction** of *point and click on the tool-specific action* using mouse input modality. In response, the system invokes the **InteractionAction** *open diagram view and show diagram* using composite static output modality for displaying the model.

3.3 Application of M ^{3} framework

To study the applicability of the presented M^3 framework we used it during the design of a visual modeling tool for UML class diagrams for a tablet device. Tablet device allowed multimodal interaction by the hardware enablers of the touch surface, microphone, and multimedia speakers.

The interface of modeling tool allowed the user to work on models using touch and speech input modality.Table 3.2 present the flow of events for *create class* use case for the modeling tool. To inform the user of the current state of the system auditory output was used along with visual feedback as output modality.

In total six **Steps** were defined for the use case. In step 1 user taps on a button to start voice command input. In the second step **SystemStep** a digital volume unit meter is shown to the user to provide the feedback of the signal level in the audio of entry. In step 3 (**SystemStep**) the user is informed with an auditory feedback that the interface is ready to accept the voice input. In step 4 speaks the command *create class*. Once the user has given the speech command, and the system detects silence it notifies the user that it is done listening voice command using auditory feedback in step 5. In the last step, a new class is visually displayed in the diagram.

In this chapter, we presented M^3 framework that enables capturing multimodal interaction detail in a formal and systematic approach. As this dissertation aims at improving the usability of modeling tools for modeling and model refactoring by employing multimodal interfaces, we define requirements for multimodal interaction for modeling in the following chapter.

Chapter 4

Requirements Specification and Design Goals for Multimodal Modeling

In this chapter, we define requirements for multimodal interaction for modeling. We employ an iterative approach towards the identification of requirements. Section 4.1 describes the vocabulary for defining speech commands. Section 4.2 describes the methodology of defining touch gestures using the observations from the pilot study and expert feedback to elicit the requirements of the MiNT framework. In section 4.3 we establish a set of visionary scenarios to describe the scope of MiNT framework. These scenarios form the basis for identifying the functional and non-functional requirements of the MiNT framework.

4.1 Speech for UML Modeling

Speech as a natural mode of interaction reduces the learning curve of working with interfaces and allows the user to move around freely and use hands for different operations. Application of multimodal interfaces using speech modality is not new to software engineering and has already been employed by researchers for visual programming [LA97]. Although modern speech recognition engines offer high accuracy in recognizing the spoken commands, several factors affect speech as an input modality and its applicability in broader contexts. Peacocke et al. [PG90] identify five major factors that affect the accuracy of speech recognition as:

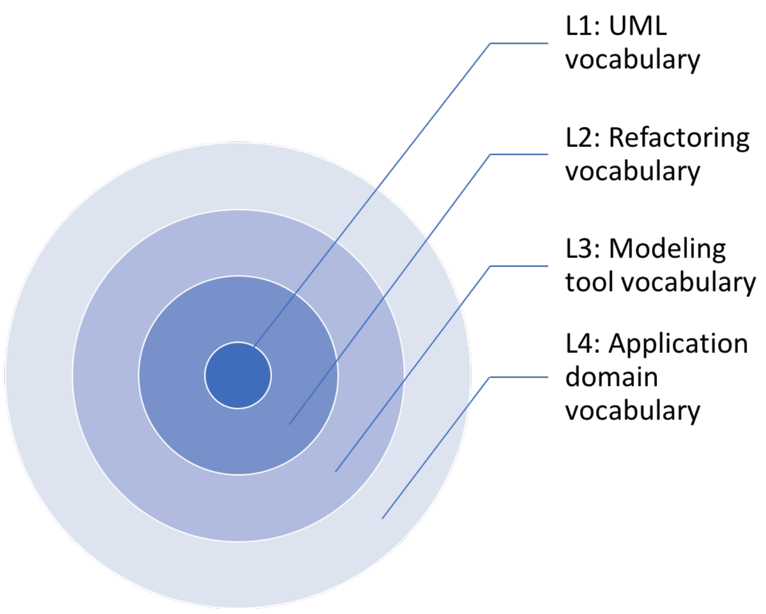
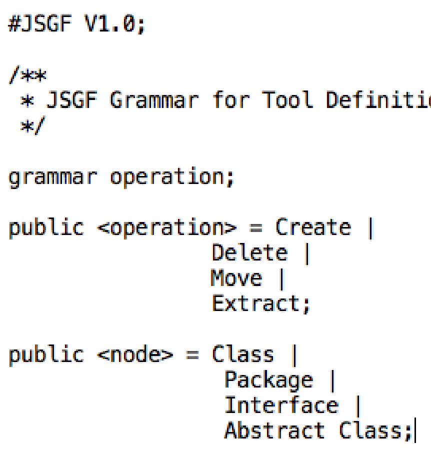

L1: UML vocabulary
L2: Refactoring vocabulary
L3: Modeling tool vocabulary
L4: Application domain vocabulary

Fig. 4.1 Vocabulary for UML modeling tools

1. *Isolated words:* Speech with isolated words (short silences between the words) is easier to be recognized as word boundaries are difficult to be found in the continuous speech which is common to natural human communication.
2. *Single speaker:* Speech from a single user is easier to recognize than from a variety of speakers. Therefore, most speech recognition systems are speaker-dependent and produce different level of accuracy depending on the speaker's pronunciation.
3. *Vocabulary size:* Size of the vocabulary of words to be recognized influences the recognition accuracy. Large vocabularies are more likely to contain ambiguous words than small vocabularies making them difficult for speech recognition engines.
4. *Grammar:* the grammar of the recognition domain defines the allowable sequence of words from the vocabulary. In a tightly constrained grammar number of words that can follow any given word is smaller. Speech recognition is more accurate with tightly constrained grammar since it reduces the search space of words that can occur in a sequence.
5. *Environment:* Background noise from the environment can significantly affect and lower the speech recognition accuracy.

When using speech as an input modality user can invoke commands to interact with the modeling tool or the model. Different layers of vocabulary are combined by the user



```
#JSGF V1.0;

/**
 * JSGF Grammar for Tool Definitions.
 */

grammar operation;

public <operation> = Create |
                Delete |
                Move |
                Extract;

public <node> = Class |
                Package |
                Interface |
                Abstract Class;
```

Fig. 4.2 JSGF grammar for speech recognition of simple spoken commands

to formulate speech commands for modeling and model refactoring. In figure 4.1 we present the vocabulary identified for spoken commands. Layer 1 represents the elements of the UML vocabulary which is defined in the UML meta-model. UML meta-model defines concepts such as classes, attributes, and methods. While UML allows extending the vocabulary by using UML profiles for domain-specific needs, otherwise the vocabulary remains unchanged.

Layer 2 define the refactoring vocabulary based on Fowler et al. [FB99]. This layer represents the body of words forming the model refactoring use cases such as **extract superclass**, or **push down attribute**. In appendix A we describe the use case considered for model level refactoring in this dissertation. Layer 3 consists of the tool-specific vocabulary that enables formulation of speech commands common to modeling tools. This vocabulary can contain words describing various user interface components such as diagram editor, canvas, or the words commonly associated with diagramming tool such as order, group, style, and layout. Layer 4 contains the words describing application domain concepts.

The UML vocabulary contains only a small set of words. The modeler is required to have the knowledge of the vocabulary. It makes the UML vocabulary suitable for a command-and-control style of speech input. A command-and-control speech input reduces the complexity of speech recognition by defining a rigid syntax that constrains possible speech input [ROR01]. Command and control interaction allows users to interact with a system by speaking commands restricted to a set of pre-defined phrases with frames for substitution of variables. For example, *create class* ***class-name***. In this example, ***class-name*** can be substituted with a word from the application domain vocabulary.

A command-and-control style language can be defined using JSpeech Grammar Format (JSGF) [1]. Figure 4.2 shows a sample grammar to recognize speech commands such as *create class*, *extract interface*, *delete package*.

In the next section, we describe the pilot study to investigate the feasibility of touch gestures for UML modeling.

4.2 Gestures for UML Modeling

Touch surfaces have been used widely to mimic the physical interactions of manipulating physical objects on flat surfaces. Mignot et al. [MVC93] studied the use of speech and gesture and found that gestures are effective for simple and direct commands, whereas speech is more effective for abstract commands. For example, gesture allows interacting with visible objects directly, whereas using speech user can interact with objects that can be described using natural language or are not visible for interacting directly. Wu et al. [WB03] describe a multi-finger and whole hand gestural interactions for tabletop surfaces. Ringel et al. [RRS$^+$04] report a set of gesture for sharing documents in a co-located tabletop environment.

Wobbrock et al. [WMW09] highlight that gestures defined by the interface designers can be helpful for early investigations, but they are not reflective of the user behavior. Cafaro et al. [CLK$^+$14] describe two approaches to defining gestures as top-down approach and bottom-up approach. In a top-down approach, the designer defines the gesture that the user can perform to interact with the system. Whereas, in the bottom-up approach users participate in the design and development of interaction gestures.

We employ a mixed approach for identifying touch gestures for modeling since application context could impact the users choice of gestures [WMW09]. These gestures were presented to the modeling experts in the pilot study along with modeling and model refactoring use cases that could be performed using specific gestures. The experts were asked to provide their opinion on the mapping of a gesture to modeling use case, reassign a gesture to a different use case if they found the mapping unintuitive, or propose a new gesture. A causality approach was also employed in which we presented a source model and a target model after

[1] https://www.w3.org/TR/jsgf/

refactoring, and asked the experts to suggest a gesture they will use naturally to perform the transformation.

A set of general characteristics for interacting with touch interfaces was identified to form the basis for identifying and describing gestures. We identify three attributes for these interactions as:

- **Naturalness** defines the general characteristics of the interaction. An interaction can metaphorically represent the similarity or closeness to performing the interaction with a physical object naturally. For example, a swipe gesture is metaphorically similar to clearing a tabletop of an unwanted object by the motion of hand forcing the object away.
- **Context** attribute defines whether the user interaction requires any contextual information to perform the use case. For example in a context-dependent interaction user needs first to select two classes to create an association. Context-independent interactions do not require any additional information to perform the use case (for example creating a class or package on the canvas).
- **Complexity** describes how modalities are employed for defining a user interaction. In a sequential interaction, a user can provide input in one modality followed by another modality. An example of sequential interaction is when a user first points at an object and then speaks a command. In a parallel interaction, two or more modalities can be employed at the same time. For example, while pointing at the object, the user simultaneously speaks a command.

4.2.1 Results

The expert review sessions were conducted with modelers from software engineering background. Since they were familiar with touch interfaces, and modeling and model refactoring process, they can easily follow the mapping of different gestures to different modeling use case. Table 4.1 presents the list of all the identified gestures and their mapping to modeling and model refactoring use cases.

To perform an operation such as the **create class**, the modelers preferred the *double tap gesture* on an empty area within diagram canvas. Some modelers used *drag and drop gesture* as an alternative since they are familiar with performing this operation if they have already used a CASE-tool user interface. **Add attributes** and **add methods** to a class was carried out with a *drag and drop gesture*.

Use case	**InteractionAction**
Create class	One finger double *tap gesture*, or *drag and drop* class from the palette
Delete class	Once finger *swipe right*, followed by *swipe left* on existing class (similar to drawing a cross)
Create attribute or method	*Drag and drop* attribute or method from the palette
Create an association	One finger *double tap* on the source class, followed by a *line draw gesture* to the target node
Delete association	Once finger *swipe right*, followed by *swipe left* on an association (similar to drawing a cross)
Change association type	One finger *swipe gesture* on an association
Change association direction	Two finger *swipe gesture* on association
Extract superclass	Select multiple class, followed by two finger *swipe up gesture*
Extract subclass	Select a class, followed by two finger *swipe down gesture*
Merge classes	Select multiple class, followed by two finger *pinch gesture*
Collapse hierarchy	Select super class, followed by two finger *pinch gesture*
Merge subclass	Select multiple subclass, followed by two finger *pinch gesture*
Move attribute or methods	Long press touch event to select attributes or methods, followed by a *line draw gesture* to the target node
Replace attributes with objects	Long press touch event to select attributes, followed by a *line draw gesture* to the empty canvas

Table 4.1 InteractionActions with touch gestures for modeling and model refactoring use cases

To **create an association** between two classes the modelers found the *line draw gesture* from source to target intuitive. To **change the association type** the *one finger swipe gesture* on an existing association was preferred. To **change association direction**, modelers preferred the *two finger swipe gesture*. A *swipe right* gesture followed by *swipe left* gesture was used by the modelers for **delete class** and **delete associations**.

For **extract superclass** use case a *two finger swipe up gesture* was considered to be appropriate. For the **extract subclass** use case modelers described *two finger swipe down gesture* as the most intuitive gesture interaction.

The **merge classes** refactoring use case was found to be similar to grabbing two or more physical objects and forcing them to collide and form a single entity. Modelers preferred a *two finger pinch gesture* on selected classes for this use case. Modelers found the*pinch gesture* also useful for **collapse hierarchy** and **merge subclasses** use case. Wu et al. [WSR+06] also highlights that gesture reuse reduces the number of gesture primitives that a user must learn. Though, for both the interactions, the context was found to be different. **Collapse hierarchy** should work only if a single class with multiple subclasses is already selected for the interaction. For the **merge subclasses** use case, the context requires two or more subclasses with a common superclass to be selected during the interaction.

4.3 Requirements for Multimodal Modeling

For the elicitation of the requirements, we use a scenario-based process and start with the identification of the actors of the system. The identified actors represent the users of the MiNT framework.

The **Modeler** is the main actor for multimodal modeling and model refactoring offered by the MiNT framework. The framework should support an individual and collaborative working style.

Developers of new multimodal interfaces must be able to use existing modalities, as well add new modalities to the MiNT framework. The framework should allow developers to define and configure multimodal fusion.

4.3.1 Visionary Scenarios

Visionary scenarios are used to define the usage of *a future system'* and are instances of use cases [BD10]. In this section, we describe two scenarios from the modelers perspective, and two scenarios from the developers perspective.

Scenario 1: Multimodal Modeling

Bean is a software consultant and frequently employs modeling for collaboration and communication. After returning from a client meeting, he realizes that he is not very well aware of the application domain concepts. He decides to do some brainstorming on his new interactive whiteboard. He launches multimodal modeling tool, on his machine, which is connected with the interactive whiteboard and walks in front of the board. He touches the whiteboard and speaks *create class savings account* using his Bluetooth headset paired to his smartphone. Next, he adds some more classes, attributes, methods, and associations to enrich his model. As he is not sure about certain part of the model, he draws a circle gesture to highlight the elements and asks Dodo who is sitting on the other side for her feedback. Dodo puts her headset on and walks in front of the whiteboard. She realizes that Bean added multiple common attributes to **savings account** and **checking account** classes. She immediately employs the multimodal input by selecting the two classes using touch input and speaks the command *extract superclass account*. Immediately a new class appears on the canvas with the name **Account** and all the common attributes from the existing classes. A generalization association is created from the **Account** class to the existing classes.

Scenario 2: Collaborative Multimodal Modeling

John and Amy are working on a ride sharing application. Since it is semester break and sunny weather, they decide to walk to the nearby lake and work on the design of the system. Once at the lake, they pull out their tablet devices and start the multimodal modeling application. John already has an existing design, which they can reuse. They connect their tablets using Wi-Fi Direct [All13] and set up an ad hoc peer-to-peer communication channel to share the model. They identify two refactoring tasks for the shared model: 1) refactoring the model by merging a few classes, 2) getting rid of a nested class hierarchy. John decides to merge the classes, while Amy agrees on reducing the hierarchy in parallel. John performs a multimodal interaction by selecting the classes using touch input and a pinch gesture. Immediately, all existing classes are replaced with a new class that contains all attributes and methods from the old classes. The changes done by John are immediately visible on Amy's tablet.

Scenario 3: Integrating new Modality

Paul is developing a hand tracking glove as input enabler for virtual reality application. He realizes that his glove can also be used to interact with the models freely. Since his glove is based on an inertial measurement unit (IMU) and does not restrict movement, he can turn any white surface into a canvas with the help of a projector. He uses the MiNT framework to prototype a modeling tool integrating motion input from his glove, and speech input. He defines the multimodal integration of these two modalities in the framework and then sets out a user study to evaluate his prototype.

4.3.2 Requirements

Based on the pilot studies, literature review, and identified visionary scenarios we describe the functional requirements (FR), and non-functional requirements (NFR) for the MiNT framework.

FR1: Multimodal Modeling

The MiNT multimodal modeling framework must support interactions using different input and output modality enablers such as touch surfaces like tablets and whiteboards, motion controllers such as leap motion, and speech input. It must allow modelers to work with gestures within the context of modeling and model refactoring tasks identified in section 4.2.

FR2: Collaboration

The MiNT framework must enable multiple users to interact with a model at the same time in collaborative modeling. Collaboration is an important aspect of modeling for having a shared understanding and representation of the modeled phenomena. Collaborative modeling sessions facilitate communication among stakeholders for *knowledge transfer and transformation*.

Collaborative modeling on a single user interface is affected by input device-specific factors. For example, some interactive whiteboards do not support multi-touch. Hence only one modeler can interact with the surface at a time. Similarly, single tablet devices are too small for two modelers to perform interaction using touch gestures at the same time. The MiNT framework must allow collaboration using multiple input devices.

FR3: Multimodal integration and customization

A key challenge in the development of multimodal interfaces is the high cost in time of implementing a multimodal interface from scratch [JS07]. The MiNT framework must support quick prototyping of multimodal interfaces for modeling tools. It must allow the easy integration of different modalities for multimodal fusion. The MiNT framework must provide interfaces for the developers to make their tool specific customization. For example, the developers can configure the mapping between gestures and the resulting actions on the model.

NFR1: Real-time Multimodal Integration

A key non-functional requirement for multimodal interfaces is the low latency during the integration of different modalities, and appropriate instantaneous user feedback. A slow system response can lead to confusion and frustration among the users. For example, if the system takes too much time recognizing a speech command, without providing any visual feedback of the process the user might get confused. In face, the MiNT framework must process different modalities in real-time and perform the fusion efficiently without any noticeable delay.

NFR2: Extensibility of Modalities

In pilot studies, we identified the need to be able to switch between the various speech recognition engines, for example, to allow input in different natural languages. The MiNT framework must enable the developer of a multimodal modeling interfaces to replace or extend the existing components of the MiNT framework. For example, it must be possible for the developer to extend the existing gesture vocabulary by adding new letter or shape strokes gesture [CZ07]. It must also allow developers to add entirely new modality enablers such as Kinect motion sensor, or Myo [2] wearable gesture and motion control.

[2]https://www.myo.com/

Chapter 5

MiNT Framework and Applications

This chapter describes analysis and design of the MiNT framework for multimodal modeling requirements identified in the previous chapter. Section 5.1 presents the analysis object model of the framework. In section 5.2 we identify the design goal. The architecture of the framework is presented in the section 5.3. Section 5.4 describes the multimodal integration in MiNT framework. In section 5.5 we describe a reference implementation of the MiNT framework using Eclipse Papyrus modeling editor for UML Class diagram. Section 5.6 describes another implementation of the MiNT framework for a multimodal modeling tool on a tablet device.

5.1 Analysis Model

Figure 5.1 shows the object model of the framework. *InputEvent* is the core abstraction of the object model. It describes any interaction of the user that can invoke an action on the model. We categorize events in two groups based on the nature of their usage.

Selection Events

TouchEvent, *MotionPointerEvent*, and *MouseInputEvent* are user interaction events that trigger the selection of UML model elements on the user interface. *TouchEvent* is specific to surfaces allowing user input in the form of touch interaction, for example, from a tablet device or an interactive whiteboard surface. *MotionPointerEvent* is triggered when the user points at a UML model element on the screen using a motion sensor. Both event types facilitate selection of multiple model elements at the same time. *MouseInputEvent* is triggered when the user interacts using a mouse as input device.

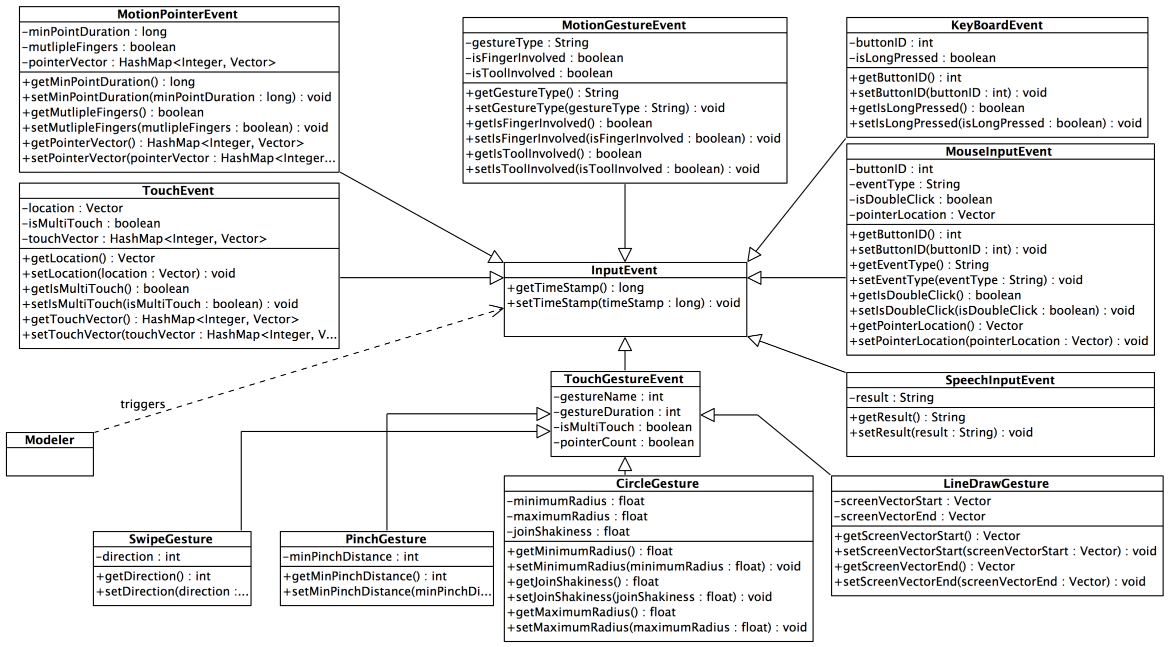

MotionPointerEvent
-minPointDuration : long
-mutlipleFingers : boolean
-pointerVector : HashMap<Integer, Vector>
+getMinPointDuration() : long
+setMinPointDuration(minPointDuration : long) : void
+getMutlipleFingers() : boolean
+setMutlipleFingers(mutlipleFingers : boolean) : void
+getPointerVector() : HashMap<Integer, Vector>
+setPointerVector(pointerVector : HashMap<Integer...
MotionGestureEvent
-gestureType : String
-isFingerInvolved : boolean
-isToolInvolved : boolean
+getGestureType() : String
+setGestureType(gestureType : String) : void
+getIsFingerInvolved() : boolean
+setIsFingerInvolved(isFingerInvolved : boolean) : void
+getIsToolInvolved() : boolean
+setIsToolInvolved(isToolInvolved : boolean) : void
KeyBoardEvent
-buttonID : int
-isLongPressed : boolean
+getButtonID() : int
+setButtonID(buttonID : int) : void
+getIsLongPressed() : boolean
+setIsLongPressed(isLongPressed : boolean) : void
TouchEvent
-location : Vector
-isMultiTouch : boolean
-touchVector : HashMap<Integer, Vector>
+getLocation() : Vector
+setLocation(location : Vector) : void
+getIsMultiTouch() : boolean
+setIsMultiTouch(isMultiTouch : boolean) : void
+getTouchVector() : HashMap<Integer, Vector>
+setTouchVector(touchVector : HashMap<Integer, V...
InputEvent
+getTimeStamp() : long
+setTimeStamp(timeStamp : long) : void
MouseInputEvent
-buttonID : int
-eventType : String
-isDoubleClick : boolean
-pointerLocation : Vector
+getButtonID() : int
+setButtonID(buttonID : int) : void
+getEventType() : String
+setEventType(eventType : String) : void
+getIsDoubleClick() : boolean
+setIsDoubleClick(isDoubleClick : boolean) : void
+getPointerLocation() : Vector
+setPointerLocation(pointerLocation : Vector) : void
triggers
Modeler
TouchGestureEvent
-gestureName : int
-gestureDuration : int
-isMultiTouch : boolean
-pointerCount : boolean
SpeechInputEvent
-result : String
+getResult() : String
+setResult(result : String) : void
CircleGesture
-minimumRadius : float
-maximumRadius : float
-joinShakiness : float
+getMinimumRadius() : float
+setMinimumRadius(minimumRadius : float) : void
+getJoinShakiness() : float
+setJoinShakiness(joinShakiness : float) : void
+getMaximumRadius() : float
+setMaximumRadius(maximumRadius : float) : void
LineDrawGesture
-screenVectorStart : Vector
-screenVectorEnd : Vector
+getScreenVectorStart() : Vector
+setScreenVectorStart(screenVectorStart : Vector) : void
+getScreenVectorEnd() : Vector
+setScreenVectorEnd(screenVectorEnd : Vector) : void
SwipeGesture
-direction : int
+getDirection() : int
+setDirection(direction :...
PinchGesture
-minPinchDistance : int
+getMinPinchDistance() : int
+setMinPinchDistance(minPinchDi...

Fig. 5.1 Object model of interaction modalities for multimodal interaction (UML class diagram)

Manipulation Events

The *TouchGestureEvent* represents gesture events that the user can perform while interacting with the interface. The *SwipeGesture* event has a *direction* attribute to distinguish if the event was triggered in an up, down, left, or right direction. The *SpeechInputEvent* describes speech input as interaction modality. Gestures from the motion sensor are defined as *MotionGestureEvent*. Finally, we identify traditional keyboard-based user interaction as *KeyboardEvent*.

The list of events described above is considered as the core events. Extending the framework with new modalities that require new events is accomplished by adding a new subclass of *InputEvent* with its modality-specific attributes. For example, developers using the MiNT framework can introduce a new subclass of *InputEvent* for stroke events to recognize shape and symbol gestures from touch events.

The class diagram shown in figure 5.2 presents the multimodal integration capability. The *DefinitionManager* object contains all the *InteractionDefinitions* defined for interacting with the UML models. A set of *DefinitionManager* objects can be instantiated for different diagram types by specifying the *modelEditorID* attribute. For each *InteractionDefinition* an *actionHandler* must be provided, which gets notified when the specific interaction oc-

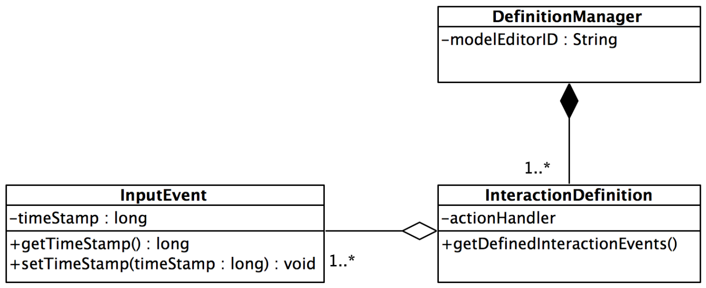


Fig. 5.2 Object model of interaction definition (UML class diagram)

curs. Whether the interaction is unimodal or multimodal is determined by the number of *InputEvents* in the definition.

5.2 Design Goals

The design goals of the MiNT framework refine the nonfunctional requirements of the framework (see section 4.3.2), as well as feedback from the expert modelers and observations made during the pilot studies.

Flexible multimodal fusion

Users can use the interaction modalities in parallel or sequentially. For example, a user can first perform a touch gesture and then speak a command, or use both modalities at the same time. This allows the developers to design and develop the new combination of modalities (multimodal fusion) for user interactions in multimodal interfaces. MiNT framework must provide this flexibility of modality integration.

Human readable interaction definition

Developers must be able to create new unimodal or multimodal interaction definitions in a human-readable markup language format. These definitions should be independent of the modeling language implemented by the modeling tool.

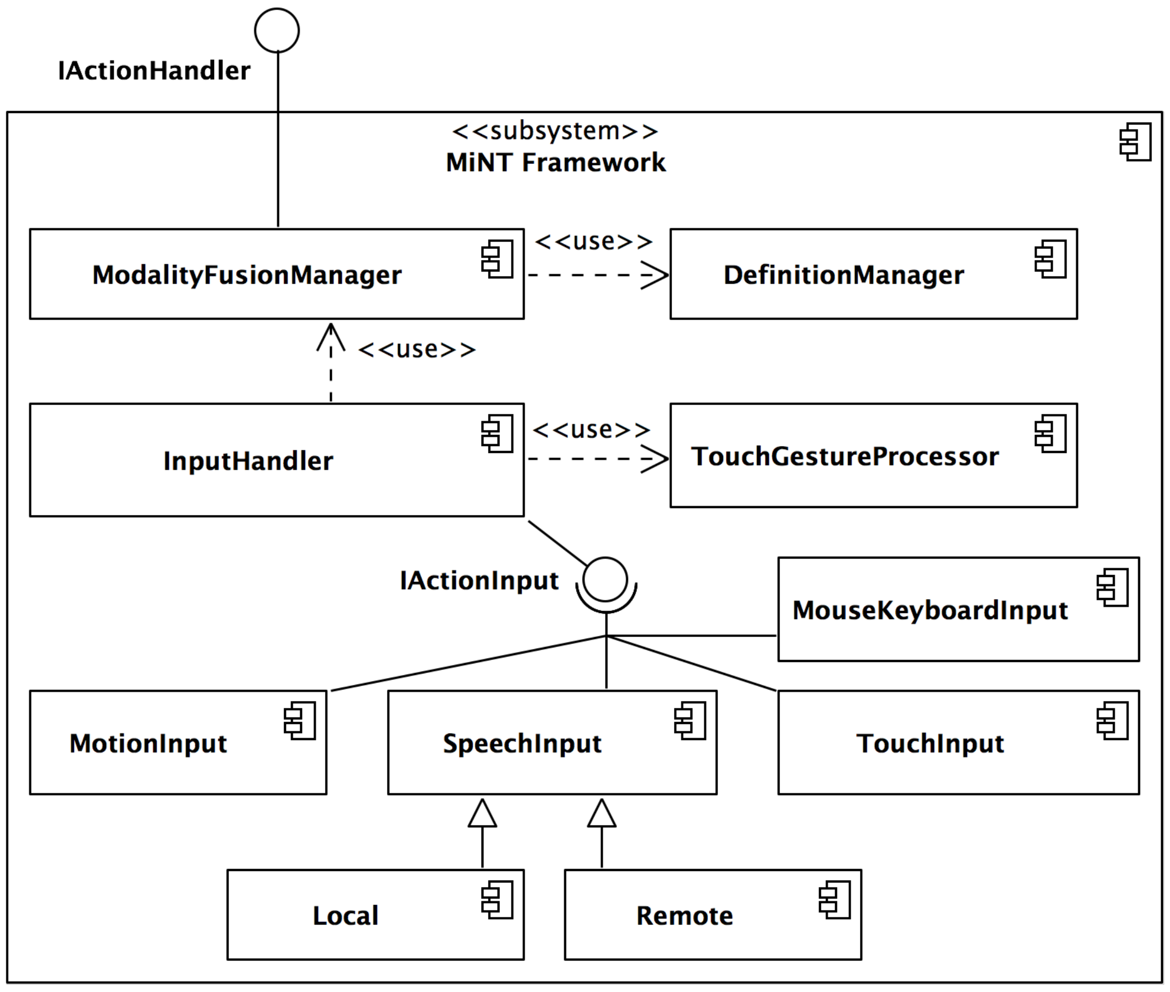


Fig. 5.3 MiNT framework architecture (UML component diagram)

5.3 Architecture of MiNT

MiNT is based on a component-based architectural style to enable low coupling and high cohesion. Figure 5.3 shows the architecture of the MiNT framework. The component-based architecture provides the possibility to replace an old component with a new component as long as the new component conforms to the prescribed behavior in the system architecture [HC01].

The *SpeechInput* component is responsible for enabling speech interactions. MiNT framework provides two different versions of the *SpeechInput* component. A local speech recognition engine and a component that allows integrating cloud-based speech recognition systems such as Nuance Cloud services [1], or Google cloud speech API [2]. Local speech recognition engines provide freedom of instrumenting the speech recognition process by adding custom grammar and vocabulary. The *SpeechInput* component provides a simple socket based server to enable any remote client to send the speech recognition results. This allows using the microphone of a mobile phone for speech input when the user interacts with an interactive whiteboard surface and is physically away from the desktop microphone (enables freedom of movement).

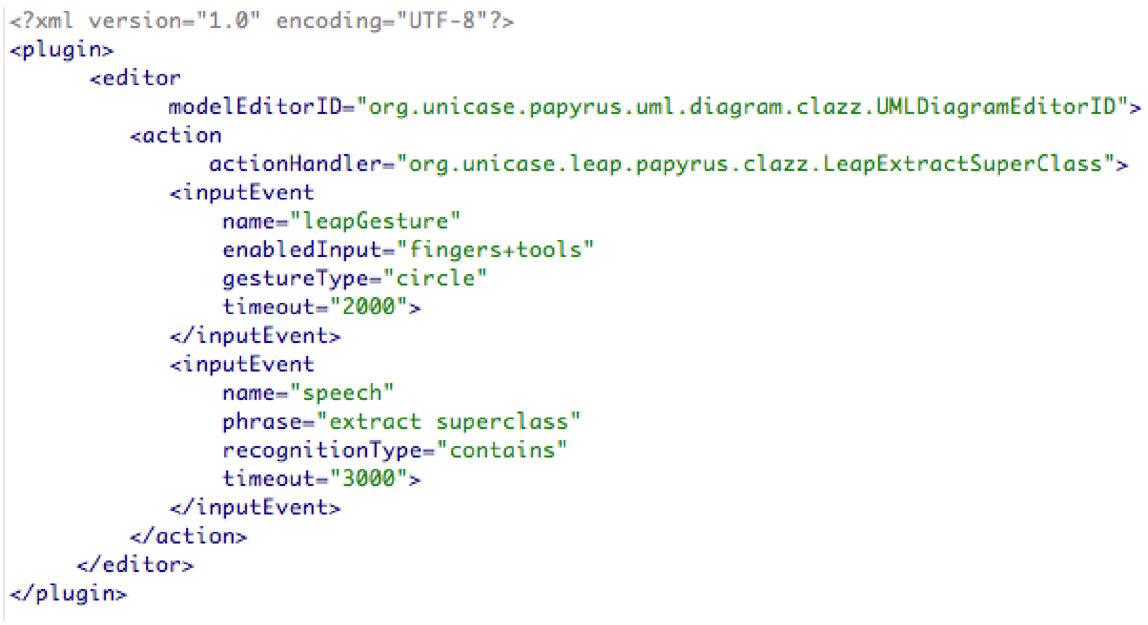
The *TouchInput* component is responsible for capturing the event stream from a touch surface such as tablet device or an interactive whiteboard surface. Touch interaction on any touch surface produces screen coordinate information that is captured by this component. This information is translated from the coordinate system of touch surface to the coordinate system being used by the model editor to point or select an object in the model editor.

MotionInput component allow integrating hand, finger, or body movement data to interact with the models. When a user interacts using hardware enablers such as Leap motion sensor, Kinect motion sensor or Myo, motion data is produced by these sensors and made available to the *Motion Input* component. This motion information is used for pointing and selection of model elements after translating to the coordinate system of the model editor.

The *MouseKeyboardInput* component captures mouse click, mouse movement events along with keyboard events from the system to enable traditional mouse-keyboard of interactions.

Developer of multimodal interfaces can add new modality components by instantiating the *IActionInput* interface of the framework, and defining the new class of *InputEvent* objects.

[1]https://developer.nuance.com
[2]https://cloud.google.com/speech/



```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
      <editor
            modelEditorID="org.unicase.papyrus.uml.diagram.clazz.UMLDiagramEditorID">
         <action
               actionHandler="org.unicase.leap.papyrus.clazz.LeapExtractSuperClass">
            <inputEvent
                name="leapGesture"
                enabledInput="fingers+tools"
                gestureType="circle"
                timeout="2000">
            </inputEvent>
            <inputEvent
                name="speech"
                phrase="extract superclass"
                recognitionType="contains"
                timeout="3000">
            </inputEvent>
         </action>
      </editor>
</plugin>
```

Fig. 5.4 Example modality integration definition for MiNT framework

Modality components notify *InputHandler* component through the *IActionInput* interface if a particular event is triggered while user is interacting using the modality (see FR3 4.3.2). The *InputHandler* component decouples the various input modalities from how they are processed for multimodla fusion. Additionally, the *InputHandler* component uses the *TouchGestureProcessor* component to identify the gesture pattern in the stream of touch data received from *TouchInput* component.

The *ModalityFusionManager* component receives the input event data captured by the modality components, and processes them incrementally to identify any unimodal or multimodal interactions as defined in the *DefinitionManager* component. The developer can define the fusion of modalities using a simple Extensible Markup Language (XML) format (described in next section). Thus, separating modality integration from tool-specific actions, allowing use the same definition for different contexts (see FR3 4.3.2). For example, a circle gesture can be utilized for selecting multiple objects in a UML class diagram editor, as well as for creating a use case in a use case diagram.

If the user interaction events match the interaction definitions, the corresponding *actionHandler* is notified by the MiNT framework. Figure 5.4 presents an example of an interaction definition using XML format. The actionHandler is notified when the user draws the circle using motion input, followed by the speech command *extract superclass*. The timeout property defines the validity duration of the event. Once the timeout period is over the event is discarded. The event from another modality should be triggered within the specified timeout period of a modality to invoke a multimodal interaction.

5.4 Multimodal Integration

In this section, we describe the attributes available for defining multimodal integration. Table 5.1 presents different modalities, and the associated attributes to formulate the interactions. This definition can be extended by adding new attributes for existing or new modalities.

While interacting using two or more modalities, there could be a slight delay in the input from the user. For example, the user can make a circle gesture first and then say *merge classes* using speech interface. Similarly, the user can first say the command and then perform the gesture. Johnston et al. [JBV$^+$02] approach this issue by introducing a short timeout to distinguish unimodal from a multimodal interaction. Hence, we introduce a common attribute across all the modalities, namely *Timeout*. Timeout allows specifying the duration after which the input event is discarded by the input processor component, providing the flexibility of temporal integration of input lattices.

Motion Gesture Input properties allow model editors to receive gesture events from the MiNT framework. Motion input consists of four different gestures (1) Screentap, (2) Keytap, (3) Circle, and (4) Swipe. Screentap gesture is a forward tapping movement of a finger or tool. Keytap is considered when a downward movement of finger or tool is recognized, similar to pressing a button on the keyboard. Circle gesture is performed by a circular motion of finger or tool in the air, while a swipe gesture is triggered by a linear motion of the finger or tool in any direction. Additionally, the developer can specify whether the user should make the gesture with the finger, or tool to be notified of the event.

Keyboard Input properties allow receiving key pressed events. Hold property triggers the event in case of the specified key was pressed for a specified time interval. This is synonymous to long press events on touch interfaces. Similarly, *Mouse Input* is a set of properties to configure and listen for events performed using the mouse as an input medium. For mouse events, *Button* attribute defines the button for the event, while it is also possible to constraint for double click events using the defined button.

Touch Input properties allow receiving input events derived from the touch interaction by the user. Different gestures are identified and extracted by the gesture processor component for example swipe, pinch, and line draw.

Table 5.1 Properties to define interaction definition (unimodal or multimodal) with MiNT Eclipse

Input Modality	**modality attribute**	**Description**
Motion Gesture Input	Gesture Type	Gesture types such as screentap, keytap, circle, or swipe identified in the user interaction
	Enabled Input	Conditional attribute to define if the gesture is performed using a finger, tool (e.g. a pen or pencil), or using both.
	Timeout	The time span defined in milliseconds that can pass before a timeout occurs and the event is discarded.
Keyboard Input	Key	Key that needs to be processed for the action to be executed.
	Hold	If the key should be in the pressed state to invoke the event
	Timeout	The time span in milliseconds that can pass before a timeout occurs and the event is discarded.
Mouse Input	Button	Mouse button that needs to be clicked to invoke the event
	Double Click	Flag denoting if the user needs to double click with the specified button
	Timeout	The time span in milliseconds that can pass before a timeout occurs and the event is discarded.
Touch Gesture Input	Gesture Type	Identified touch gesture such as pinch, swipe, circle, or line draw
	Timeout	The time span in milliseconds that can pass before a timeout occurs and the event is discarded.
Speech Input	Phrase	Phrase that needs to be recognized in the output of speech recognition output
	Recognition Type	Pattern of phrase in the identified text for example exact, start, contains, end
	Timeout	The time span in milliseconds that can pass before a timeout occurs and the event is discarded.
Pointing	Fingers	Boolean attribute to denote if the finger motion input is used for mouse pointer
	Tools	Attribute to configure if the tool (e.g. a pen or pencil) detected in motion input is used to update the mouse pointer location
	Touch input	If the input from touch surface (interactive whiteboard, or mobile device) is used to update the pointer
	Multiple pointers	Multiple fingers or multi-touch data (if available) should be visualized

Speech Input properties are essential if the interface integrated speech as an input modality. In *extract superclass*, or *add attribute*. Once the speech recognition result is found to be containing the phrase, Input processor components check for the recognition type attribute. It could be specified to have the speech command completely or partially match the phrase. Or, contain the phrase at the start or end of the speech command spoken by the user, to be considered as a valid condition for the event.

Pointing attributes aims at configuring how the desktop pointer is manipulated in response to the user input with the help of pointing device. If the motion input of fingers and tools, both are used to enable pointer movement. Similarly, touch events on an interactive whiteboard can result in pointer location update. Additionally, multiple pointers property allows interacting using multiple fingers or a multitouch surface. This property enables visualizing and capturing multiple users interaction, and thus useful for collaborative modeling on a shared machine.

In the next section, we present how the model editors can provide multimodal integration capabilities in their user interfaces by describing the integration with an Eclipse-based modeling framework.

5.5 MiNT Eclipse

Eclipse [3] provides several existing modeling and visualization frameworks and technologies (for example; Graphical Modeling Framework (GMF), and Graphiti) for rapid prototyping of domain-specific modeling CASE-tools. As a cross-platform integrated development environment (IDE) used as a platform of choice for researchers and developers to design and develop tool support for domain-specific modeling notations for desktop operating systems. Thus we realized that it is the most appropriate platform to show the applicability of MiNT framework, and thus we developed a reference implementation named MiNT Eclipse.

MiNT Eclipse follows the plug-in concept offered by Eclipse [CR08]. Plug-ins are software components in Eclipse and facilitate extending applications with additional features using plug-ins. Plug-ins provide extension points to expose functionality from one component to another thus enabling loose coupling between components. Components can describe extension points as a contract that any other plugin which is interested in extending the functionality must implement. Since the component specifying the extension point knows nothing about the component which implements the extension, it allows developers to extend

[3]https://eclipse.org

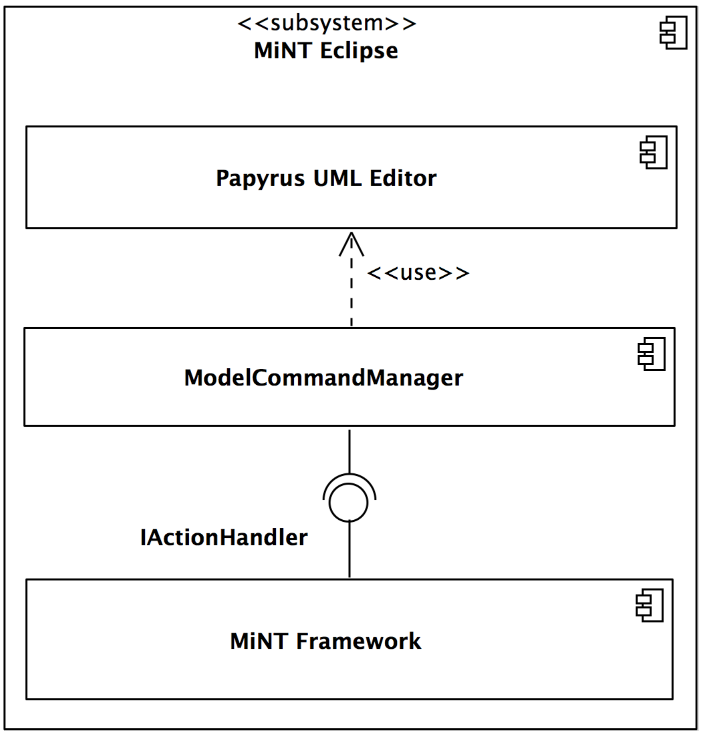


Fig. 5.5 Architecture of MiNT Eclipse reference implementation (UML component diagram)

or introduce new functionalities easily. Since Eclipse runtime realizes the OSGi specification [All15], plug-ins can be started or stopped at run-time.

To demonstrate the applicability of a multimodal interface for modeling, we integrated MiNT framework into Eclipse-based Papyrus modeling tool [LTE$^+$09] to implement MiNT Eclipse. Papyrus provides editors for UML diagrams following the UML 2.0 specification and supports SysML for model-based system engineering.

Two sets of use cases were identified for MiNT Eclipse with the focus on basic modeling, and for model refactoring of UML class diagrams. All the use cases employ touch input as the means of pointing or selecting elements in the diagram. The speech input is used to trigger diagram manipulation commands.

Figure 5.5 presents the integration of the Papyrus UML in the MiNT framework. The *ModelCommandManager* component in the MiNT Eclipse is responsible for providing handlers to act on the notification of MiNT framework events. For the basic modeling use case, we defined the handlers (see table 5.2) using the interaction model of M^3 framework. The table provides a list of modeling use cases, the InteractionAction associated with the use case, and an example speech command in command-and-control format (bold texts are representative of words from the domain vocabulary and are changeable). Action

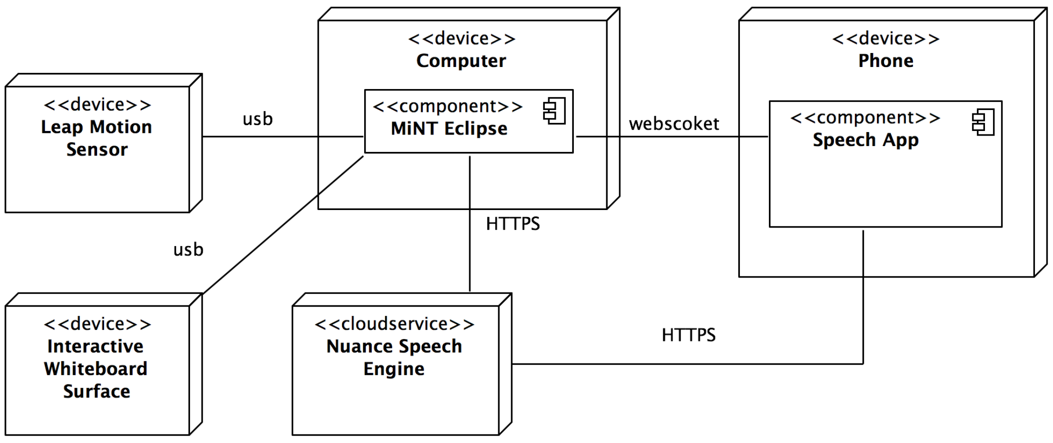


Fig. 5.6 Deployment diagram of MiNT Eclipse (UML deployment diagram)

handlers instantiate tool-specific commands to perform the changes on the model. For model transformations, the low-level commands are chained together to perform complex operations. For example, extract superclass handler first instantiates a command to create a class in the model, next, a command is executed to pull up common attributes from the subclasses to the new class, and finally, a command to pull up common methods to the new class is executed.

The model refactoring use cases are presented in the table 5.3. All the refactoring use cases were taken from the Fowler's book *Refactoring: Improving the Design of Existing Code* [FB99]. While this book primarily focuses on the code refactorings, some refactorings are also applicable for the model. Appendix A contains detail description of each of the model refactoring use cases.

MiNT Eclipse allows users to work on models using touch input, motion input, speech input and traditional keyboard and mouse input. Figure 5.6 shows the deployment diagram of MiNT Eclipse. Leap motion sensor data is transferred to the *Motion Input* component of MiNT framework using USB connection. Leap motion allows capturing hand and finger movements without the need of physical contact while the user interacts in the air above the hardware sensor. Additionally, leap motion identifies gestures such as circle gesture, or swipe gesture, from the movement of hand and finger, and makes it accessible through the leap motion SDK.

Touch interaction from interactive whiteboard is received using USB connection by the *Touch Input* component of the framework. MiNT Eclipse allows the user to invoke speech command using the microphone of the computer as well as using *Speech App* on the phone. Recorded audio is sent over HTTPS protocol to Nuance cloud services for recognition. Recognized

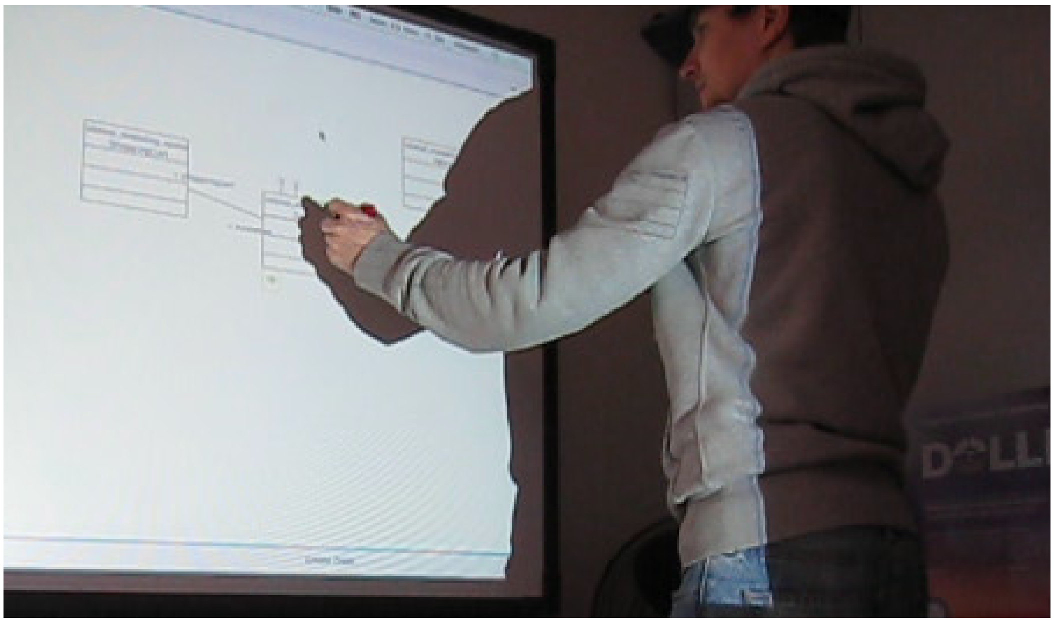


Fig. 5.7 Multimodal interaction using interactive whiteboard and speech

text is sent to the MiNT Eclipse from the phone over web-socket. A local speech recognition component based on CMU Sphinx is provided for offline usage.

Figure 5.7 shows a user interacting with the MiNT Eclipse interface using touch and speech input. In the next section, we describe MiNT Mobile implementation.

5.6 MiNT Mobile

MiNT Mobile is a standalone Android application using the MiNT framework to enable the use of touch, speech, and touch gestures for working with models using multimodal interaction.

The *ModelCommandManager* component is responsible for providing the action handler and instantiates a command for model manipulation. Additionally, the command manager allows undo-redo operations on a model. An implementation of the memento pattern is employed for this purpose [Gam95]. Commands once executed, modify the underlying UML model, and notify *DesignerView* subsystem to update the graphical user interface.

The *DesignerView* maintains a list of nodes and associations and on update requests, the *ViewItem* of each element available in *UMLModel* subsystem to render themselves on the canvas provided by DesignerView. Since the DesignerView provides a generic implemen-

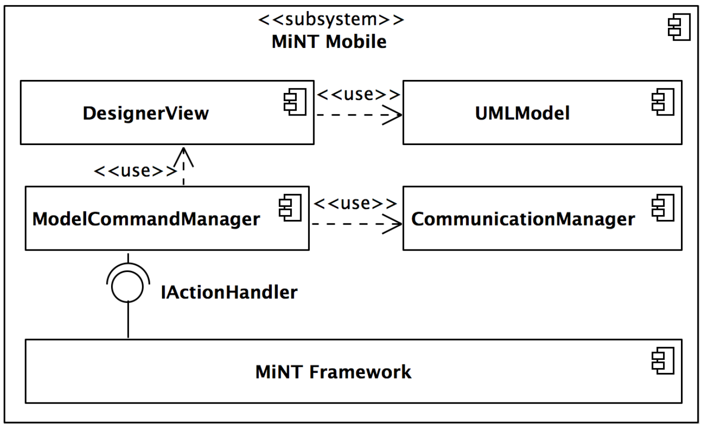


Fig. 5.8 Architecture of MiNT Mobile reference implementation (UML component diagram)

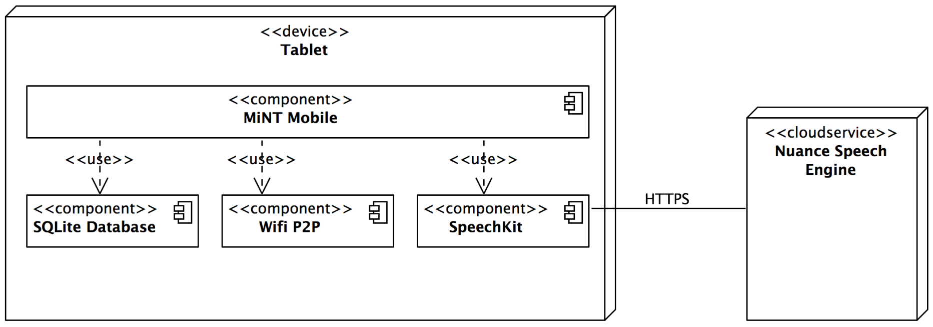


Fig. 5.9 Deployment diagram of MiNT Mobile (UML deployment diagram)

tation and is coupled with *UMLModel* component, it provides the extensibility to provide support for new diagram types without changing the underlying implementation. Similarly, the underlying model could be easily extended to support additional UML Class diagram elements such as package or interface.

A main requirement of the MiNT framework is to facilitate real-time modeling and collaboration between modelers (see FR2 4.3.2). The *CommunicationManager* component establishes a peer-to-peer communication across the different devices. Modelers can share their models with other peers. The *CommandExecutor* synchronizes all the commands and propagates them to the connected peers to have the same state of the model on all the peers. Wi-Fi Direct [All13] allows a *Collaboration* component on Android devices to communicate with each other directly without the need of any internet connection. This component enables collaborating on UML models as well as diagram sketches.

Figure 5.9 shows the deployment diagram of MiNT Mobile implementation. MiNT Mobile uses SQLite database[4] component provided by the Android application framework to save the

[4]https://sqlite.org/

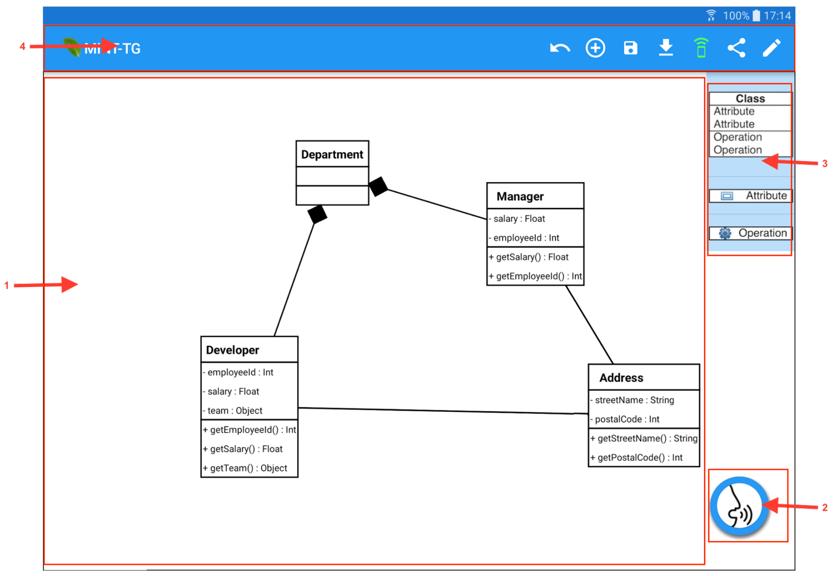


Fig. 5.10 MiNT Mobile user interface

models locally. Peer-to-peer communication for collaboration is enabled by the use of *Wifi P2P* component of the Android framework. MiNT Mobile uses the Nuance Cloud services for the speech recognition. Speech recognition is enabled by the *SpeechKit* component that is part of the nuance mobile SDK.

In the next section, we present the user interface of MiNT Mobile, realized within the *DesignerView* component.

5.6.1 MiNT Mobile User Interface

The user interface of MiNT Mobile consists of the four parts shown in Figure 5.10. The central area (annotation 1 in the figure) is the canvas view to visualize all the model elements. The canvas view is continuously monitored by the *GestureObserver* subsystem to extract touch gestures from the touch events. Annotation 2 in the figure highlights a floating action button, which allows users to notify the *SpeechKit* component to start listening for the speech commands. As soon as the silence is detected, the *SpeechKit* subsystem sends the recorded audio to a cloud-based automatic speech recognition engine. Additionally, the user can manually press the button to send the recorded audio for speech recognition.

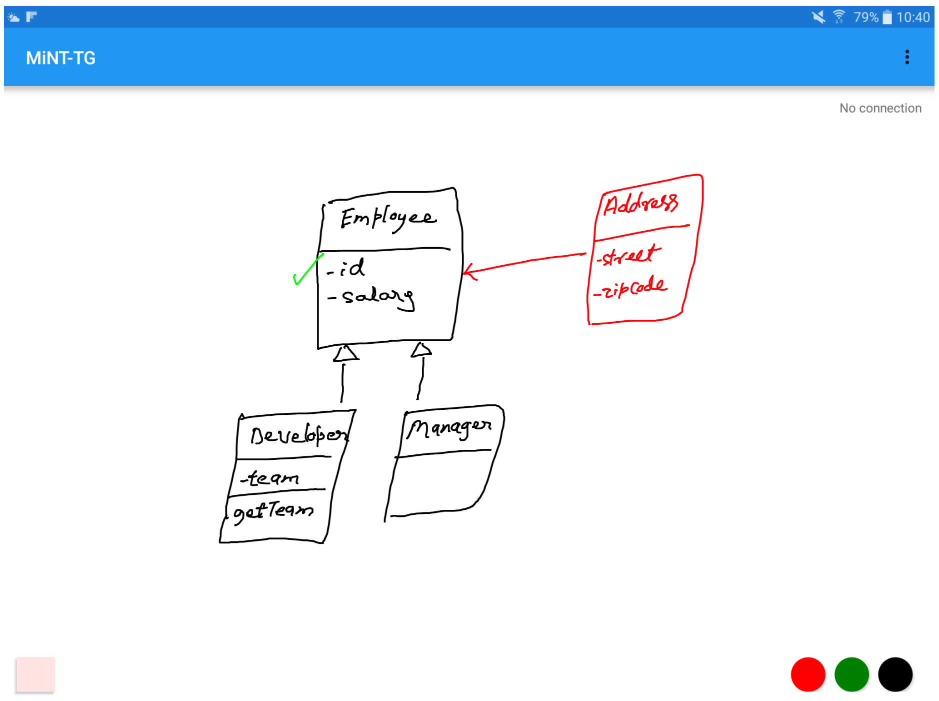


Fig. 5.11 MiNT Mobile: Informal modeling using touch sketches

MiNT Mobile enforces the Java naming convention [5] and automatically performs the necessary changes to the letter case, on the speech recognition input as and when applicable. For example, a speech command *add attribute first name* results in an attribute with the name *firstName*. Additionally, it performs the low level consistency check on the model. For example, a class can have only one instance of an attribute or method with the same signature.

Annotation 3 as shown in the figure is a small palette with the model items available for the current diagram type. Users can drag and drop elements from the palette onto the canvas. Attributes and methods require the drop location to be a class in the canvas for a valid model update. For frequently used tool-specific features, the interface provides an action bar on the top (annotation 4 in the figure). Actions such as undo-redo model change, create a new model, save or load model, enable-disable continuous speech mode (detailed in the section 5.6.2), share the model with peers, and collaborative informal modeling, can be accessed quickly and without interfering with the model currently open in the canvas view. Figure 5.11 presents an informal model created using MiNT Mobile.

Another critical component of the MiNT Mobile user interface is the model edit dialog. While for lightweight and rapid modeling touch, speech, and touch gestures are sufficient, to

[5] http://www.oracle.com/technetwork/java/codeconventions-135099.html

add more detail (formalization) in the model, a user can open the dialog with a long press gesture on a class node. This dialog allows adding, remove, and update attributes or methods. Additionally, one can specify visibility and data types for attributes and methods, which is immediately reflected in the canvas view.

MiNT Mobile support the similar set of use cases as identified for Papyrus UML class diagram editor for a combination of touch and speech input. Additionally, MiNT Mobile allows user interaction with touch gestures, as identified in the pilot study (reported in 4.2.1). Touch input is primarily for selection of model elements, whereas touch gesture is employed for actions resulting in model manipulations. MiNT Mobile provides three different alternatives to interact with the user interface for naming and renaming use cases (1) Touch input on the software keyboard, (2) Speech input similar to the InteractionActions for Papyrus UML class diagram editor, (3) Continuous Speech Mode (detailed in the next section).

5.6.2 Continuous Speech Mode

A majority of user interactions while working with models are directed towards the naming and renaming of model elements using domain-specific terminology. A software keyboard on touch interfaces allows a user to provide the input by keystroke. While developing MiNT Mobile, we observed that a significant amount of time was spent by the user to type the name of model elements once they have created structural modifications using the combination of touch and touch gestures. We, therefore, provided speech as input modality to reduce the amount of touch interaction for this use case.

Observations made during the evaluation of hypothesis 1 revealed that a majority of subjects reported fatigue caused by two reasons; First, speaking long commands continuously and second, the reduced speech recognition accuracy for longer speech commands. Thus, we needed to devise an efficient and intuitive approach of utilizing speech in combination with the other two modalities employed by MiNT Mobile. To solve this problem, we introduce **Continuous speech mode**.

Continuous speech mode is a context aware speech input enabler. Once the mode is activated from the action bar of the MiNT Mobile interface, the *CommandExecutor* component starts to monitor all commands that create new model elements. As soon as such a creation command is captured to be invoked from touch or touch gesture modality, the *SpeechKit* component is notified to listen for speech input. If the speech lattice is captured within a

Use case	InteractionAction	Example speech command
Create class	Point the location on canvas and use speech command	*Create class* ***Employee***
Delete class	Point on an existing model element and use speech command	*Delete* or *delete class*
Rename class	Point an existing model element and use speech command	*Rename class* ***account***
Create associations	Select two model elements and use speech command	*Add association*
Delete associations	Select two model elements with an existing association and use speech command	*Remove association*
Create attribute or methods	Point to an existing class and use speech command	*Add attribute* ***name***
Delete attribute or methods	Point to existing class containing attribute or method and use speech command	*Delete attribute* ***email***
Rename attribute or methods	Point to existing class containing attribute or method and use speech command	*Rename attribute* ***name*** *to* ***first name***

Table 5.2 InteractionActions for Papyrus UML Class diagram editor with touch and speech input

Use case	InteractionAction	Example speech command
Extract super class	Select one or more model elements and use speech command	*Extract super class **Employee***
Extract sub class	Select one model element and use speech command	*Extract sub class **Bus***
Merge classes	Select two or more model elements and use speech command	*Merge classes*
Collapse hierarchy	Select a model element with subclasses and use speech command	Collapse hierarchy
Pull up attribute or method	Select model element with the attribute or method and use speech command	*Pull up attribute **email***
Push down attribute or method	Select model element with the attribute or method and use speech command	*Push down attribute **email***
Pull up common	Select two or more subclasses and use speech command	*Pull up common*
Move attribute or method	Select source and target model element with the attribute or method and use speech command	*Move attribute **email***

Table 5.3 InteractionActions for model refactoring actions

certain timeout period, it is wrapped in a rename command and executed on the newly created model elements. This allows integration of touch, touch gesture and speech modalities in a seamless and natural interaction. For example, a user can perform a *drag and drop gesture* to create a class, attribute or method, and in parallel to performing the gesture speak the name of the element. Similarly, after performing an **extract superclass** use case using two finger *swipe up gesture*, a user can name the new class using speech. To rename attributes or methods, a modeler can perform a long press touch event on an existing element, and input a new name using speech.

Continuous speech mode, apart from making the interaction more natural, reduces the number of interactions needed to start the speech command manually or to type a name using the software keyboard.

In the next chapter, we describe the experiments conducted to evaluate the hypothesis of this dissertation.

Chapter 6

Evaluation Multimodal Modeling on Interactive Whiteboard

In this chapter, we describe the experiments conducted to study the applicability of multimodal interface employing an interactive whiteboard surface and speech as an input modality for modeling and model refactoring. MiNT Eclipse reference implementation was used for the controlled experiment. In section 6.1 we establish the hypothesis for the controlled experiment. Section 6.2 describe the independent and dependent variables. Section 6.3 and section 6.4 present the profile of the experiment participants and detail the experiment setup. Section 6.6 presents the results of the controlled experiment. Finally, we discuss the findings of the experiment and reports threats to the validity of the results.

6.1 Context

A new or a complex interface with hidden menus and actions can introduce additional cognitive load to recall, remember, and discover actions while working on complex modeling tasks. A common reason for increased cognitive load is the abundance of information to be processed at a given time. Complex task increases the number of information units in working memory that interacts with each other. These information units are required to be processed simultaneously to fulfill task objective. For example driving from one point to another requires several information units such as knowledge about navigation path, traffic and weather conditions, distance from the car/object in front, and obvious knowledge about driving and controlling the vehicle. Cognitive load theory differentiates between three types of cognitive load in the design and evaluation of instructions:

- ***Intrinsic:*** It is the inherent level of difficulty caused by the structure and complexity of any given content. Intrinsic cognitive load depends upon the number of informational units a learner needs to hold in the working memory to comprehend the information [PCS02].
- ***Extraneous:*** The cognitive load imposed by the manner in which information is presented to the user and by the working memory requirements of the instructional activities is termed as an extraneous cognitive load. This load is a form of overhead that does not contribute to an understanding of the content and reduces the number of cognitive resources available to process the intrinsic and germane load [CS91].
- ***Germane:*** Germane cognitive load represents the load induced by the learner's effort to process and comprehend the material [SVMP98]. In the modeling task, this load can be attributed to the mental model created by the modeler.

We believe multimodal interfaces employing speech could be effective in reducing the cognitive load by allowing the modelers to naturally interact with the interface, as oppose to traditional unimodal interfaces. Thus, enabling modelers to improve their efficiency during modeling and model refactoring tasks.

We formulate null hypothesis for the controlled experiment:

> **$\mathbf{H1_0}$:** Modelers using multimodal interface utilizing an interactive white-board surface and speech input have no improvement in their efficiency than the modelers using traditional unimodal modeling interface during modeling and model transformation activities.
>
> Corresponding alternative hypothesis is:
>
> **$\mathbf{H1_A}$:** A multimodal interface utilizing an interactive whiteboard and speech input improves the *efficiency* of modelers during modeling and model transformation activities.

Roland Brunken [BPL03] categorize various approaches to measure cognitive load in two categories as, *objectivity* and *causal relation*. ***Objectivity*** describes whether the approach uses ***subjective*** data or ***objective*** observations for the evaluation of cognitive load. While ***Causal relation*,** classifies approaches based on the type of relation of the phenomena observed by the measure and the actual attribute of interest. For example, a direct link exists between cognitive load, the difficulty of the content, change in efficiency of task completion and error rate. For the evaluation of our hypothesis, we consider a combination of subjective and objective approaches.

6.2 Variables

In an experiment, the independent variable is the variable that is varied or manipulated by the researcher, and the dependent variable is the response that is measured. An independent variable is a presumed cause, whereas the dependent variable is the presumed effect.

Independent variables:

- **Setup:** It is a binary variable capturing whether the subject is using the multimodal interface employing speech and touch interaction or any general UML modeling tool as the baseline approach. For this experiment, we employed Visual Paradigm [1] as the baseline modeling tool.
- **Application domain knowledge:** It is not always the case that the modelers are very well aware of the application domain being modeled. Frequently modelers take the input of application domain experts in a separate session and try to produce a structured model subsequently in a different session either alone or with a group of modelers. Further, there is considerable evidence that domain specific knowledge is a key factor distinguishing experts from novices within the context of problem-solving [Swe88]. Unfamiliarity with the application domain can increase cognitive load and affect performance in explorative tasks. To this end, we categorize subjects with **low**, **medium** and **high** awareness of the application domain. Low represents awareness of some domain-specific terminology, medium represents being able to associate entities within the domain, and high represents subjects with the working experience.
- **Modeling frequency:** Basic modeling knowledge is a pre-requisite for the recruitment of the subjects. The subjects are classified in three categories based on their modeling frequency i.e. **low**, **medium**, **high**. Low represents a subject group with modeling frequency within the range of one twice per month (also could be considered returning or infrequent users). A medium is a group of subjects using modeling tools and techniques one twice per week (frequent modelers). High-frequency subjects are the one practicing modeling and model transformation within the context of their work almost every day (very frequent modelers).

[1]https://www.visual-paradigm.com

Dependent variables:

1. Objective Measures:

    - **Time to complete:** Cognitive load can be correlated with the ***efficiency*** of the user and relate to the performance. We define 'efficiency' as the amount of time required to complete a task. If the user's cognitive load increases either due to the unfamiliarity with the tools interface or due to complex navigations within the tool, he is supposed to take more time to complete his task.

    - **Number of errors:** As the complexity of tasks increases, users can make errors, which explicitly affect the *efficiency* variable. We make an observation to see if any correlation exists between the complexity of modeling task, tool interface, and the number of errors. The number of errors can affect time to complete as modelers have to spend additional time to correct the mistakes. We only consider errors that occur or lead to change in the model. The following two different error types are identified:

        - **E1:** Unintentional errors occur due to the lack of tool usage knowledge or the unwanted behavior of the tool resulting in change within the model. Such errors require modelers to undo their changes and find an alternative approach to accomplish his task. For example, the tool creates self-association for a class when the user is trying to create an association from one class to another, or creating/deleting an element unintentionally while trying to create something else (could be attributed to either of the two gulfs defined by Norman).

        - **E2**: These are the error specific to a ***modeling task*** for example creating an irrelevant class, adding attribute and methods in wrong objects, or adding unnecessary association. Such errors could be attributed to an incomplete mental model of the application domain, solution domain, cognitive load or individual attentional process. Frequently if the modeler can spot the error, they rectify their mistakes with the combination of undo-redo or copy-paste mechanism. For example cut and paste attribute to the correct class, or delete it from the wrong class and recreate in the target class.

2. Subjective Measures:

    - **Self-reported difficulty of materials:** is a subjective measure that aims at capturing the cognitive load from the perspective of the modeler or the tool user. The difficulty of the materials could be attributed to the difficulty of the task itself, individual competency, or attentional processes.

- **Self-reported difficulty using interface for a certain task:** describes the usefulness of any tool interface from the modelers perspective to accomplish their task.

6.3 Subjects

Modeling knowledge is an essential prerequisite for this study. 13 subjects were recruited with basic modeling experience using UML. We believe that if the subjects are aware of modeling tools and techniques, they should easily be able to start working with either of the two interfaces. Subjects were randomly assigned into two setup treatment groups. In this experiment, we did not consider creating a balanced group by expertise as our objective was to explore and understand the generality of the hypothesis.

6.4 Setup

The experiment required the subjects to work on two different tasks, modeling, and model transformation. Tasks were identified and improved in pilot sessions with the help of several volunteers before the final experiment to have a more clear, unambiguous, and realistic representation of the task description along with supporting material (see appendix B.2).

Task 1	Modeling : Online Rental Platform
Task 2	Model transformation (Refactoring and Refinement) : Rental Company

Table 6.1 Task Categorization

Task 1 required (as shown in Table 6.1) the subjects to work on the creation of a simple application domain model representing the domain elements and their associations. Task description was formulated as natural language requirements instead of instructional format or bullet points. Providing the description in instructional format can influence or interfere with the subjects natural thinking process of breaking down the task into logical steps. Task 2 was aimed at model transformation employing model refactoring and model refinement to extend an existing model. Task description was formulated as transformation steps in an instructional format. The application domain of the tasks was 'Online Rental Platform' and Organizational hierarchy of a 'Rental Company.' These domains were chosen with the assumption that the subjects should have some awareness of the domain, model elements, and their associations.

For this experiment, MiNT Eclipse speech recognition system was not instrumented to use a domain vocabulary. An unconstrained vocabulary allowed us to have a realistic observation for the applicability of a speech interface in a global software engineering project setup, where stakeholders with different speech accents can interact with the system. In such scenarios, it is not possible to either train the speech recognition system with a customized acoustic model for every stakeholder or constraint the vocabulary with predefined words from a domain as domain vocabulary tend to be rather flexible and evolving with the software project.

6.5 Procedure

The experiment was divided into three parts. In the first part subjects working with the multimodal interface, treatment was given a short introduction on how to interact and work with the multimodal interface and the infrastructure setup. Additionally, subjects were provided with a list of speech commands supported by the MiNT Eclipse interface for reference. Subjects assigned to baseline treatment were given a basic tool usage training in case subjects asked for it explicitly. All subjects reported that they have already worked with the baseline tool earlier. In the second part of the experiment, subjects were provided with the first task description.

Once the subject has finished reading the description, they notified the instructor as soon as they were ready to start working on the task using the tool interface. Subjects were required to use **think aloud** methodology working on the tasks. Thinking aloud during problem-solving means that the subject speaks out whatever thoughts come to mind while performing the task [VSBS$^+$94]. After completing the task, subject notified the instructor. Subsequently, they were provided with the second task description along with the existing model, and the subject performed the task in the same format as of the first task. The second part of the experiment was video recorded for further analysis with the consent of the subject. In the third and last part of the experiment, subjects were asked to fill out an online questionnaire. Further, instructor reviewed subjects responses with them to get a clear understanding of their selected choices. Subjects were additionally encouraged in the session to share their insight and observations made during the experiment.

6.6 Experiment Results

6.6.1 Collected Data

In total, 13 subjects voluntarily participated in the experiment. At random subjects were assigned to baseline treatment, which was a familiar modeling tool for all the participants, or the MiNT Eclipse treatment. Out of six subjects assigned for baseline treatment total three had Masters (either pursuing or completed) as minimum education qualification, while educational qualification of remaining three was Ph.D. (either pursuing or completed). Seven subjects were assigned to MiNT Eclipse treatment, with two subjects having Bachelors and Masters each as their qualification while remaining five had Ph.D. (either pursuing or completed). Detailed information on the subjects and the collected data during the experiment is presented in the following section.

ID#	Setup	Education	Domain Familiarity (Task 1)	Domain Familiarity (Task 2)
1S	baseline	Master	medium	low
2E	baseline	Master	medium	high
3M	baseline	Master	medium	medium
4J	baseline	Ph.D	high	high
5A	baseline	Ph.D	medium	medium
6D	baseline	Ph.D	medium	medium
7D	MiNT Eclipse	Bachelor	low	medium
8K	MiNT Eclipse	Master	medium	medium
9S	MiNT Eclipse	Ph.D	high	medium
10Z	MiNT Eclipse	Ph.D	medium	medium
11C	MiNT Eclipse	Ph.D	high	high
12D	MiNT Eclipse	Ph.D	high	high
13J	MiNT Eclipse	Ph.D	low	medium

Table 6.2 Collected data from the experiment (part 1)

Setup, education level of subjects, self-reported domain familiarity for modeling task, self-reported domain familiarity for model transformation task is summarized in Table 6.2. Table

ID#	**Modeling Frequency**	**Time to Complete in seconds (speech overhead) (Task 1)**	**Time to Complete in seconds (speech overhead) (Task 2)**	**#E1**	**#E2**
1S	Once-twice per week	511	717	-	4
2E	Once-twice per month	768	780	3	-
3M	Once-twice per week	292	580	9	1
4J	Once-twice per week	347	281	4	2
5A	Once-twice per week	284	663	11	-
6D	Once-twice per month	646	512	-	-
			Number of total errors	**27**	**7**
7D	Once-twice per week	369 (51)	394 (104)	2	1
8K	Once-twice per month	263 (174)	352 (40)	-	1
9S	Once-twice per month	294 (40)	370 (70)	2	-
10Z	Once-twice per week	399 (116)	427 (71)	-	-
11C	Once-twice per month	430 (49)	541 (50)	1	-
12D	Once-twice per week	367 (46)	327 (70)	3	1
13J	Once-twice per month	316 (52)	358 (16)	3	2
			Number of total errors	**11**	**5**

Table 6.3 Collected data from the experiment (part 2)

6.3 presents self-reported modeling frequency of the subjects, time to complete modeling task, time to complete model transformation tasks, the number of type one error, and the number of type two error during modeling and model transformation task. Additionally, time to complete for MiNT Eclipse section separately within bracket reports the additional time spent by the participant to repeat the speech command if the speech recognition engine was not able to provide correct recognition result on the first attempt. All the time variables are reported in seconds and recorded using stopwatch during the experiment. Further, we cross-checked time to complete variable and errors with the help of video transcribing of each experiment session.

6.6.2 Analysis

We applied *t-Test* to test on the collected data set. A t-test was selected as a statistical method since we have two sample groups and the sample size was small (13 subjects in two groups). Since there was significant speech recognition overhead for the MiNT interface, we performed t-Test twice; with speech overhead, and ignoring speech overhead, against the baseline. We reformulate our hypothesis as:

$$H1_0 : \mu_1 - \mu_2 = 0$$
$$H1_A : \mu_1 - \mu_2 \neq 0$$

	Baseline vs MiNT Eclipse (ignoring speech overhead)		**Baseline vs MiNT Eclipse (speech overhead)**	
	Task1	Task 2	Task 1	Task 2
F at $\alpha = 0.05$	11.42	6.20	10.62	5.26
$F_{critical}$ at $\alpha = 0.05$	4.38	4.38	4.38	4.38

Table 6.4 F-test for the equality of variances

To perform *t-Test* we have to identify if the variances of the two groups are equal or not. For this we perform an *F-test [SC89].* More detailed summary of the results is presented in the appendix B.4. Since $F > F_{critical}$ for every observation (see table 6.4) we conclude that the variances of the two groups are unequal. Hence, we perform Welch's t-test or unequal variances t-test to validate our hypothesis. Unequal variance t-test was conducted as it is more reliable if two samples have unequal variances and unequal sample sizes in place of regular Student's t-test or Mann–Whitney U test [Rux06].

Observations from the the two-tailed t-test are presented below (detailed summary in appendix B.5.). For every observation if $t_{Stat} < -t_{Critical\ two\text{-}tail}$, or $t_{\text{Stat}} > t_{\text{Critical two-tail}}$, we reject the null hypothesis. We use significance level (α) of 0.05 and reject the null hypothesis if the $p_{\text{two-tail}}$value (probability of finding the observed results when the null hypothesis is true) is less thanα.

Baseline vs MiNT Eclipse (ignoring speech overhead) Task 1: There is no evidence of significant improvement in the efficiency of using MiNT Eclipse setup against the baseline approach in-spite of ignoring speech recognition overhead ($-2.44 < 1.48 < 2.44$, and $p_{\text{two-tail}} = 0.18 > 0.05 = \alpha$). Thus, the null hypothesis ***$H1_0$* can not be strongly rejected** for this scenario. Although there is evidence suggesting modelers complete modeling task when using MiNT Eclipse (M=348.28, SD=59.56) over baseline approach (M=474.66, SD=201); t(6)=1.48, p=0.18 by an ***average of 126 seconds*** **less time** in the current data set.

Baseline vs MiNT Eclipse (ignoring speech overhead) Task 2: For the model transformation task ignoring speech overhead report ***improved efficiency*** over the baseline approach ($-2.44 < 2.48 > 2.44$, and $p_{\text{two-tail}}= 0.04 < 0.05 = \alpha$). Thus, the null hypothesis ***$H1_0$* can be rejected** and alternate hypothesis ***$H1_A$*can be accepted.** Data set further reveals that modelers working with MiNT Eclipse (M=395.57, SD=71.65) against baseline approach (M=588.83, SD=178.44); t(6)=2.48, p=0.04 are***faster by an average of 193 seconds***.

Baseline vs MiNT Eclipse (speech overhead) Task 1: For modeling task with speech overhead included statistical analysis concludes rejecting alternate hypothesis with α value of 0.05 ($-2.44 < 0.59 < 2.44$, and $p_{\text{two-tail}} = 0.57 > 0.05 = \alpha$). Still, in the current data set with time to recover from speech errors included, subjects achieve an ***efficiency of 50 seconds*** with MiNT Eclipse (M=424, SD=61.78) over baseline approach (M=474.66, SD=201); t(6)=0.59, p=0.57.

Baseline vs MiNT Eclipse (speech overhead) Task 2: Similarly, including speech overhead for model transformation, suggests rejecting alternate hypothesis ($-2.36 < 1.69 < 2.36$, and $p_{\text{two-tail}} = 0.06 > 0.05 = \alpha$). For model transformation task with speech overhead included MiNT Eclipse (M=455.71, SD=77.78) user ***saved 133 seconds*** on average over baseline approach (M=588.83, SD=178.44); t(6)=1.69, p=0.13.

Our statistical observation revealed that for model transformation task there was a significant improvement in the efficiency (193 seconds on average) after ignoring the speech recognition overhead, and also data sample gives reasonable evidence to support the alternative hypothesis. For the modeling task ignoring speech overhead modelers are faster by an average of 126 seconds, but the current sample provides comparatively weak evidence of finding the same observation. For the remaining two analysis with speech overhead included, current data set show efficiency of 50 seconds(Task1) and 133 seconds (Task 2) when using multimodal interface over baseline approach. Application domain familiarity and modeling frequency do not suggest any correlation to the efficiency of modelers in the current data sample.

Next, we tried to understand any correlation between the interface type and errors subjects made while working on the tasks. Average of the baseline users creating type 1 error was 4.5 errors (SD = 4.6) per user, while for the same error type MiNT Eclipse users had a mean value of 1.5 errors (SD = 1.26). For type 2 error, baseline users had an average of 1.16 (SD = 1.6), and the user of speech interface created errors with a mean value of 0.71 (SD = 0.75). Since the sample size was rather small, we considered applying any statistical model irrelevant. From the observation, we understood that in-spite of having more errors on an average baseline approach did not suffer from any major efficiency drop due to the nature of interface setup. Subjects working with baseline approach had to sit in front of the experiment computer and perform the task, which did not cause any interruption in the workflow or introduced any overhead over their normal working with a modeling tool. On the other hand, working with MiNT Eclipse required subjects to frequently adjust their view focus by moving away from the interactive surface to have an overview of the model, and subsequently move closer to interact with the surface. Additionally, speech processing naturally takes more time as subjects have first spoken the complete command than the speech recognition system produces the speech to text of the spoken audio, and finally, the identified command is executed on the model.

6.6.3 Exit Interview

After the subjects finished the modeling task with either of the two interface assigned for the experiment they were asked to assess the difficulty of the tasks, accompanying material, and the difficulty working on a particular task using tool interface (see Appendix B.3 for the statements). Subjects provided self-reported difficulty of the task and the difficulty understanding the material for the most difficult task using binary variables. There was no significant difference in being able to understand the provided material for the most difficult

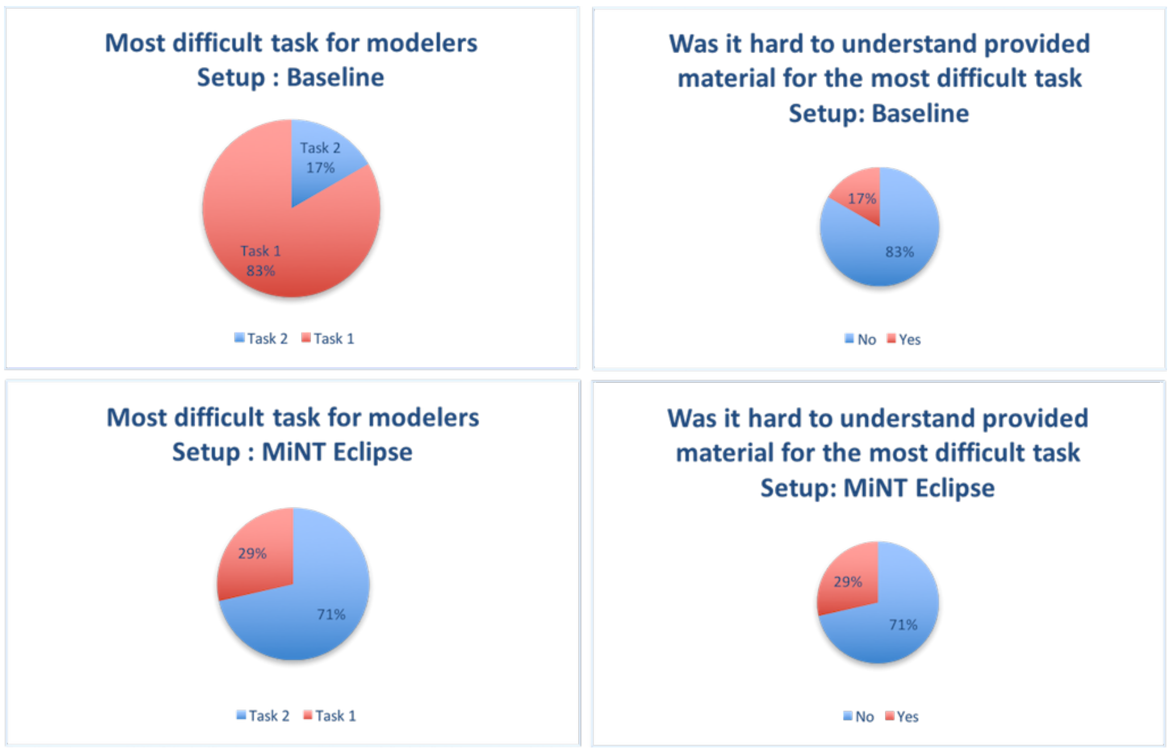


Fig. 6.1 Results: Task difficulty vs difficulty understanding the material

task among the baseline and MiNT treatment group, as most of the subjects reported that the provided material was understandable and was helpful comprehending the problem (Figure 6.1).

A significant difference was noted within the subject group regarding the most difficult task. 83% subjects working with baseline approach reported that the modeling task (task 1) was the most difficult for them, while 71% in the MiNT group agreed on the model transformation task being the most difficult (task 2). We asked the subjects to summarize the rationale behind their selections verbally. Subjects working with baseline approach reported finding model transformation easier as they could comfortably have an overview of the existing model from the comfort of their sitting position, and quickly follow the transformation steps mentioned in the provided material without much thinking. MiNT group mentioned that they had to physically change their position either to interact with the interface during model transformation task or to get an overview as they could not have the complete model in their head all the time. This introduced interruption in their problem-solving process after each transformation step.

Next, we asked the subjects to provide their subjective assessment using a Likert scale for the difficulty of performing each task using the tool interface. Figure 6.2 presents the response collected from the subjects. For the modeling task, 50% subjects from the baseline approach reported finding the tool interface introducing high difficulty, while the remaining 50% considered the difficulty to be either low or very low. 70% subjects from the MiNT group considered the interface to be introducing low or very low difficulty in task completion while remaining 30% found the difficulty level to be medium. For the model transformation task, as also coincide with the task 2 being the least difficult for the baseline group, 80% subjects reported low difficulty. At the same time for MiNT Eclipse group, we had 70% subjects reporting difficulty to be low or very low. For the MiNT group, we also had a subject reporting very high difficulty performing the model transformation task. Subject provided the explanation as *I feel nervous if being observed*, and thus was forgetting the speech command, or saying wrong or partial commands, subsequently feeling overwhelmed.

Subjects were asked if they created any mental or external to-do list to break down the problem in solution steps after reading the task description. We wanted to gain an insight into the process of identifying solution steps and if the modeling tool interaction is considered and introduced in this to-do list. All the subjects for MiNT approach agreed to have a mental or external to-do list after reading the task description, while for baseline except two subjects, remaining subjects created a to-do list. We observed during the experiment that these two subjects decided to read the task description while working on the task.

Additionally, we asked the subjects if they have anything to share with the experimenter on the tool interface usage during the experiment. Baseline users had nothing to share, although MiNT Eclipse users provide personal opinions and observations. One subject reported interactive surface combined with speech is *good for brainstorming*. It can help *verbalize thinking* over watching models being created. Another subject reported that speech *improves the speed of creating models*. *Facilitates collaboration* by allowing people to contribute to the model at the same time. One subject highlighted the issue of speech recognition overhead as *speech recognition errors interrupt the thinking/problem-solving process*. Additionally, subjects mentioned several features that could be interesting to have, for example, being able to create multiple attributes in one command (add attribute email password and phone number).

6.7 Discussion

Welch's unequal variance t-test shows that multimodal interface employing interactive surface and speech as input modality have statistically significant impact on the improvement of

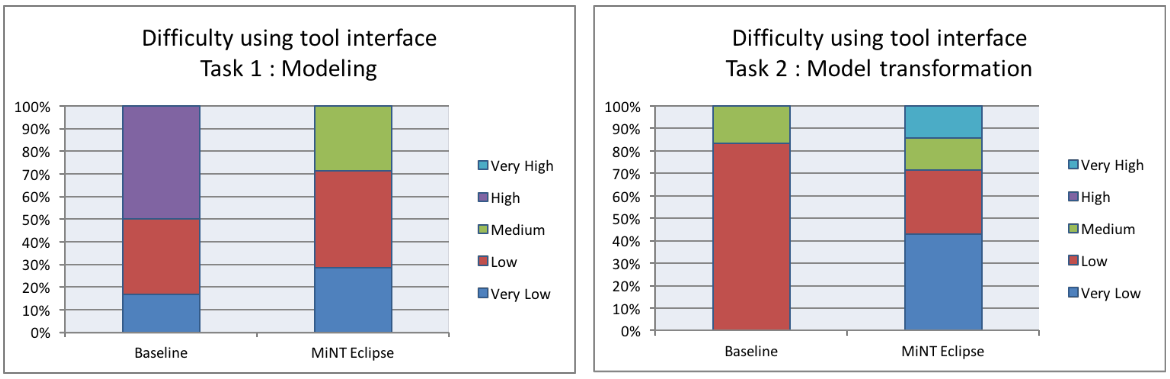


Fig. 6.2 Results: Difficulty performing task using tool interface

efficiency of model transformation. For the modeling task results for multimodal interface show improvement in the efficiency, though the probability of finding the same observation if the null hypothesis was true is higher than alpha value of 0.05 in the current data set. We think larger sample size could provide stronger evidence to reject the null hypothesis.

Based on our observation, we found that interactive whiteboard surfaces by design are more suitable for brainstorming and collaborative modeling sessions, as oppose to the efficiency of modeling. Speech as an input modality shows to improve the efficiency of modeling and model transformation and fewer unintentional errors if combined with interactive whiteboard surface, apart from additional benefits such as *allows verbalize thinking in collaborative modeling sessions,* as reported by the subjects.

Speech recognition error caused by unconstrained domain vocabulary presents a different perspective on the current state of speech recognition technologies and their applicability for a speech interface that is to be used by an international group of users with the different accent. Experiment results make it apparent that the state of *speech recognition is not ready for a global software engineering project with the requirement of explorative modeling*, where new domain-specific terms are frequently introduced during the early stages of requirements engineering process and are usually only known to domain experts. Using existing domain ontology (if available) or semi-automatically extracting frequently occurring terms in the requirements specification of projects with similar domain and using it to constrain the vocabulary of speech recognition system could help to some extent reduce the speech recognition induced errors.

The number of errors created while working with baseline approach was found to be greater than the multimodal interface, but this does not introduce any noticeable delay in the working time of modelers. We believe modelers can more quickly resolve the errors in the baseline approach as it does not require subjects to physically switch their position and thus do not interrupt in the workflow process.

6.8 Threats to Validity

We identify following threats to the validity of the controlled experiment.

- **Size of the experiment group:** The total number of subjects recruited for this experiment was relatively small with 6 out of 13 assigned to baseline treatment while remaining seven assigned to MiNT Eclipse treatment. We think for an experiment of this nature with multiple independent and dependent variables, it is still an acceptable group size to cautiously generalize the findings as an early evidence of the applicability of a multimodal interface employing interactive surface and speech input. The size of the group for baseline and multimodal treatment was also affected by the enormity of the experiment, which sometimes lasted for almost an hour, and thus prospective subjects had difficulty finding time.
- **Tool familiarity:** All the subjects of the experiment had working experience with the baseline approach, as oppose to the subject group assigned for MiNT Eclipse treatment who had to learn a new interface and start working immediately. We cautiously assume that this could have made it easier for the baseline treatment group to get started with their task and make less error. Although, this was not clearly evident from the observations of baseline users on average took more time, and had a higher error mean.

Chapter 7

Evaluation Multimodal Modeling on Mobile Devices

In this chapter, we describe the experiments conducted to study the applicability of two different multimodal interfaces in the different modeling context. MiNT Mobile reference implementation was used in the controlled experiment. In section 7.1 we establish the hypothesis for the controlled experiment. Section 7.2 describe the independent and dependent variables. Section 7.3 present the profile of the experiment participants and section and section 7.4 detail the experiment setup. In section 7.6 we present the results of the controlled experiment. Finally, we discuss the findings of the experiment and report threats to the validity.

7.1 Context

Oviatt et al. [OV96] describe the advantage of multimodal interfaces for error correction as *users can act upon good intuitions regarding the accuracy of a particular modality*. Thus dynamically user switches from one modality to another while working to reduce the modality-specific errors or the limitations of a particular modality in a certain context. We performed a follow-up controlled experiment to identify the need for a multimodal interface which is efficient, as well as suitable for different modeling contexts.

We formulate null hypothesis for the controlled experiment:

We formulate *null hypothesis* as:

H2$_0$: Multimodal interface with touch, speech, and touch gestures have no benefit over an interface employing touch and speech as input modalities for modelers.

Corresponding alternative hypothesis is:

H2$_1$: Modelers find multimodal interface employing touch, speech and touch gesture input modalities more *useful* and *practical* as opposed to an interface using only touch and speech input modalities.

7.2 Variables

Based on the hypothesis it was determined that the subjective evaluation is the most optimal approach to identifying the *usefulness* and *practicality* of either of the two interface. Though, we are also interested in understanding if the interface reported as most useful and practical is also the most efficient for modelers. In the last experiment, we already found initial evidence that interfaces with speech modality improves efficiency, provided it is not affected by errors caused by speech recognition. To this end, we identify following variables for this experiment;

Independent variables:

- **Setup:** Setup is a binary variable representing the treatment either with MiNT Mobile implementation using touch and speech interface (**MiNT TS**), or touch, speech, and touch gesture interface (**MiNT TSG**). Since we use the same tool, it neutralizes the tool expertise among subjects as they all have to familiarize themselves with the tool and its interface.

Dependent variables:

1. Objective Measures:
    - **Time to complete:** Time to complete denotes the efficiency of modelers while performing their tasks using either of the two multimodal interfaces.
2. Subjective Measures:
    - **Self-reported usefulness and practicality:** Modelers can evaluate an interface for its pragmatic, hedonic, attractiveness qualities, to determine the usefulness and practicality. We employ a combination of interview and questionnaire to collect the 'self-reported' subjective evaluation data from the subjects.

7.3 Subjects

In total 17 subjects were recruited for the controlled experiment on the voluntary basis, and the subjects performed the experiment in a one-on-one session in the presence of the experimenter. All the subjects had basic knowledge of UML Class diagram notation and experience working with CASE-tools. Every subject was required to perform the same set of tasks using both the interfaces and provide subjective feedback. Interface to start experiment was selected at random. Thus, nine subjects started with MiNT TS to complete the tasks and subsequently used MiNT TSG to perform the same task, while remaining eight subjects started with MiNT TSG, and moved on to using MiNT TS. The rationale for this randomization was to study if there was any correlation between interface order and time to complete.

7.4 Setup

The experiment required the subjects to work on two different tasks, modeling and model transformation in a similar format employed in the first experiment (see table 7.1). The first task required modelers to create a simple domain model following the description. Order tracking domain was selected such that all the subjects have at least basic familiarity with the domain entities and their relationships. The second task required subjects to transform an existing model from organizational hierarchy following the transformation guidelines provided along with an existing model. Provided material is available in the appendix C.2

Task 1	Modeling : Order tracking
Task 2	Model transformation (Refactoring and Refinement) : Organizational Hierarchy

Table 7.1 Categorization

7.5 Procedure

The experiment was divided into three parts for each interface. In the first part, subjects were introduced to the multimodal interface and asked to familiarize themselves with using the interface for five minutes. In the second part, subjects were invited to perform the modeling task, followed by model transformation task using the interface. We asked the subjects to signal experimenter when they have read the task description and want to start working with the interface. Similarly, as soon as they were done with the task, they had to signal of completion. This time duration was recorded for *time to complete* variable for both the task using the interface. In the last part, we asked the subjects to fill in a questionnaire covering questions on various aspects of usefulness and practicality of the interface. The Same procedure was repeated for the second interface after the subject has finished working on both the task using initial interface, with a minor variation. While filling in the questionnaire for the second interface, we allowed the subjects to make comparative changes in their response to the first experiment. As the subjects had their opinion about the first interface in a recorded format, and the opinion about the second interface mentally, they could easily compare both the interfaces on various factors and provide a summative response. Finally, subjects were asked to share any insight and observations within the scope of the experiment.

7.6 Experiment Results

7.6.1 Collected Data

Since to start the experiment, subjects had to work with one of the two randomly assigned interface; we capture this information as 'Initial interface' in the table 7.2. Next, for each subject time to complete modeling and model transformation task using either MiNT TS or MiNT TSG was captured using a stopwatch and recorded in seconds. We use this collected data to identify statistical significance in task completion time first among interfaces, and secondly among the order of interface selection and their impact on task completion.

In total 17 subjects participated in the experiment which lasted for 30-45 minutes. As MiNT Mobile allows working comfortably in sitting position, whereas MiNT Eclipse required subjects to physically switch places to interact, experiment infrastructure was a regular office setup with a tablet device to conduct the experiment. As presented in Figure 7.1, 59% subjects joining the experiment where either Ph.D. students or already had their Ph.D. title as the education level. Remaining 41% reported to either pursuing Masters or already have completed it. Most of the subjects reported having over three years of experience with modeling tools and techniques (76%) accumulated from either their work experience or within the course of software engineering studies. 18% subjects reported of experience in

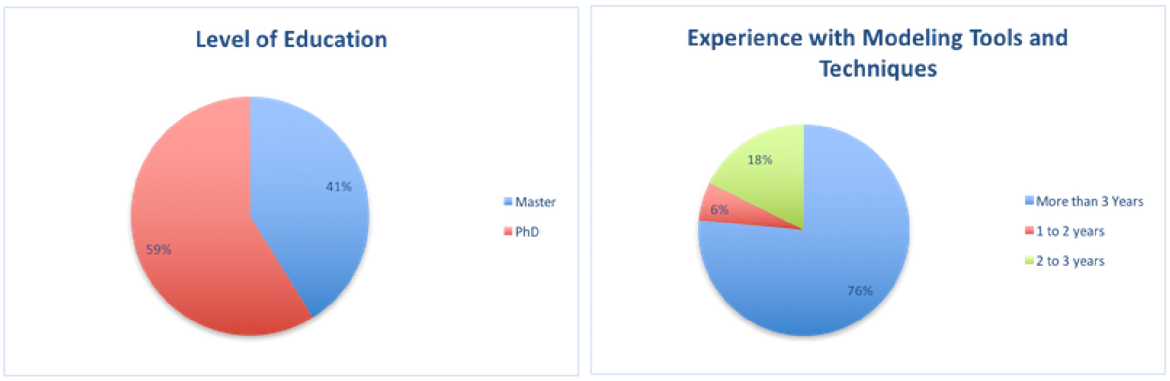


Fig. 7.1 Educational background and modeling experience of the subjects

between 2 to 3 years, while only 6% had relatively less experience in comparison i.e. 1 to 2 years.

ID#	Initial interface	MiNT TS		MiNT TSG	
		Task 1	Task 2	Task 1	Task 2
1	MiNT TS	298	235	240	153
2	MiNT TS	230	190	140	105
3	MiNT TS	251	167	130	79
4	MiNT TS	250	202	136	74
5	MiNT TS	229	242	124	75
6	MiNT TS	229	208	132	86
7	MiNT TS	161	182	92	78
8	MiNT TS	152	138	111	95
9	MiNT TS	253	212	178	94
10	MiNT TSG	148	129	166	152
11	MiNT TSG	202	204	172	115
12	MiNT TSG	274	249	238	121
13	MiNT TSG	184	182	199	155
14	MiNT TSG	188	187	160	114
15	MiNT TSG	148	183	133	134
16	MiNT TSG	220	198	232	173
17	MiNT TSG	175	172	147	116

Table 7.2 Time to complete modeling (task 1), and model transformation (task 2) by subjects

7.6.2 Analysis

We applied *paired sample t-test* to evaluate the effect of using MiNT TS and MiNT TSG on the efficiency of the modeling. Since, both the samples were collected from the same set of individuals paired t-test was identified as the most suitable technique.

Finding of the t-test is reported in APA (American Psychological Association) style (also appendix C.4);

MiNT TS vs MiNT TSG Modeling: A paired sample t-test was conducted to compare the time to complete for modeling task by subjects when using MiNT TS interface and when using MiNT TSG interface. There was a significant improvement in the efficiency of modelers to complete modeling task when using MiNT TSG (M=160.58, SD=44.41) over using MiNT TS (M=211.29, SD=46.14); t(16)=4.63, p=0.0002. Modelers took 51 seconds less when using the MiNT TSG interface to complete the modeling task.

MiNT TS vs MiNT TSG Model transformation: t-test showed significant improvement in the efficiency of modelers to complete modeling task when using MiNT TSG (M=112.88, SD=31.45) over using MiNT TS (M=192.94, SD=32.34); t(16)=7, p=0.000002. Reduction of 80 seconds was observed on an average with MiNT TSG over MiNT TS during model transformation task.

We further analyzed the data to understand if order of the interface has any affect on the modelers efficiency. Below we summarize the finding for subjects who used MiNT TS as the first interface;

MiNT TS vs MiNT TSG Modeling - Initial Interface MiNT TS: Results show that MiNT TS (M=228.11, SD=45.83) users were less efficient than MiNT TSG (M=142.55, SD=43.25); t(8)=9.59, p=0.00001 by an average of 86 seconds.

MiNT TS vs MiNT TSG Model Transformation- Initial Interface MiNT TS: During model transformation task still MiNT TSG (M=93.22, SD=24.73) users took less time than half the time of MiNT TS users (M=197.33, SD=32.57); t(8)=8.88, p=0.00002. Thus overall completing the task with an average 104 seconds less. We believe the results showed such low time for MiNT TSG subjects as by this time they were highly aware of the task by already completing is using the previous interface, as well as the interface by using it to perform the modeling task.

Next, we analyzed how the subjects performed who had to use MiNT TSG interface first to work on the tasks;

MiNT TS vs MiNT TSG Modeling - Initial Interface MiNT TSG: Results show that MiNT TS (M=192.37, SD=41.13) users were less efficient than MiNT TSG (M=180.87, SD=38.49); t(7)=1.42 p=0.19 by an average of 12 seconds. Thus, there is no significance difference between the two groups.

MiNT TS vs MiNT TSG Model Transformation- Initial Interface MiNT TSG: During the model transformation task using MiNT TSG (M=135, SD=22.44) interface subjects showed reduced time to complete (an average of 53 seconds) than while working with MiNT TS (M=188, SD=33.54); t(7)=3.28, p=0.01.

Paired sample t-test results for all the observations reveal that multimodal interface employing touch, speech, and touch gesture was helpful in reducing the time to complete for modelers. Results obtained for overall comparison between both the interfaces and the sample when subjects used MiNT TS interface first are ***statistically highly significant***. Observations reveal that subjects suffered high learning curve when they started working with MiNT TSG interface for the first task (average improvement of only 12 seconds and probability value greater than significance level). Apart from remembering touch interaction, speech commands and various gestures they had to focus on creating the model using recently acquired information. By the time they moved on to perform model transformation post-completion modeling, they were relatively more familiar with the interface, and the same is reflected in the probability value and the difference of mean for MiNT TS and MiNT TSG. We sum up the analysis results as:

> MiNT TSG allows subjects to be more efficient irrespective of whether the subjects already performed the same task using MiNT TS or not, though once gained familiarity with the task and already worked with MiNT TS subjects were highly efficient with MiNT TSG. If MiNT TSG was the first interface, subjects reported having faced information overload.

7.6.3 Subjective Evaluation

After subjects had finished with either MiNT TS using touch and speech as input modalities or MiNT TSG using touch, speech, and touch gestures, we asked them to evaluate the interface for their pragmatic qualities, hedonic qualities, and attractiveness for modeling and model transformation. *Pragmatic qualities* describe usability and usefulness of the interface. In this context, we define usability being associated with the ease of use, while usefulness is determined by the degree to which interface helps modelers expectations for frequent

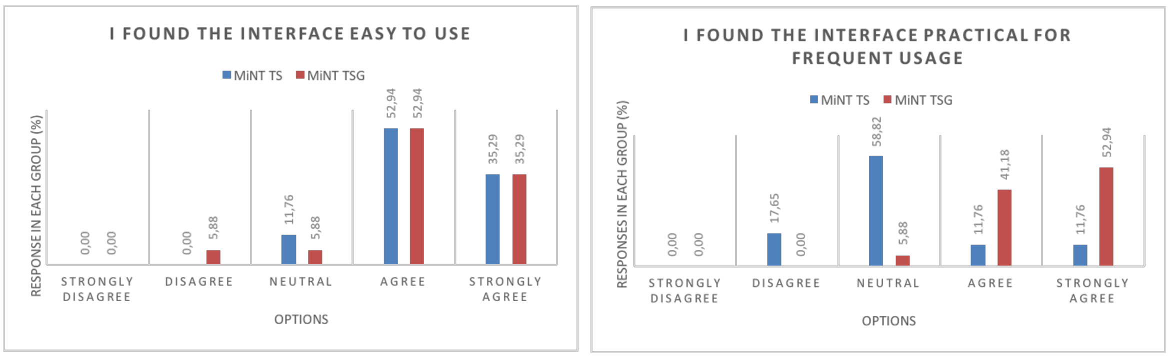


Fig. 7.2 Evaluation of pragmatic qualities for touch and speech interface against touch, speech (MiNT TS) and touch gesture interface (MiNT TSG)

usage. Subjects provided their assessment on the pragmatic attributes of the interfaces by answering following statements: (a) I found the interface easy to use; (b) I found the interface practical for frequent usage. *Hedonic qualities* denotes the aspect of interface that relates to or characterizes the pleasure, novelty, and of being able to hold interest by providing a positive experience. Following statements were focused on capturing the hedonic attributes of the two interfaces: (a) I found the interface to be creative (involving new approach towards working with models); (b) I found the interface to be captivating (being able to hold my interest). *Attractiveness* is an attribute equally influenced by the pragmatic and hedonic qualities of the interface. We use the following to statement to understand if the modelers find MiNT TS and MiNT TSG interface attractive and pleasant for frequent usage during their regular modeling and model transformation tasks: (a) I found the interface to be appealing (attractive for regular modeling and model refactoring tasks); (b) I found the interface to be pleasant (enjoyable in a sense making pleased and satisfied). Subjects provided their assessment using a five-point Likert scale ranging from strongly disagree to strongly agree.

Figure 7.2 presents the response for the pragmatic qualities of both the interfaces. For MiNT TS and MiNT TSG, an equal number of subjects responded that they either agree or strongly agree of the interface being easy to use. Out of 17 subjects, one subject (5.88% among all) disagreed for MiNT TSG being easy to use. Our observation revealed that the reason behind this response was the steep learning curve of knowing touch, speech, and touch gesture to perform the task in such short span of time. This also supports the statistical observation for **MiNT TS vs. MiNT TSG Modeling - Initial Interface MiNT TSG**, where subjects were less efficient in comparison to other observations. Next, in response to the practicality of the interfaces for frequent usage 52.94% subjects strongly agreed that touch, speech, and touch gesture makes MiNT TSG very useful. Another 41.18% subjects agreed on the practicality of MiNT TSG, and only one subject (5.88%) responded in neutral. For MiNT TS, a majority

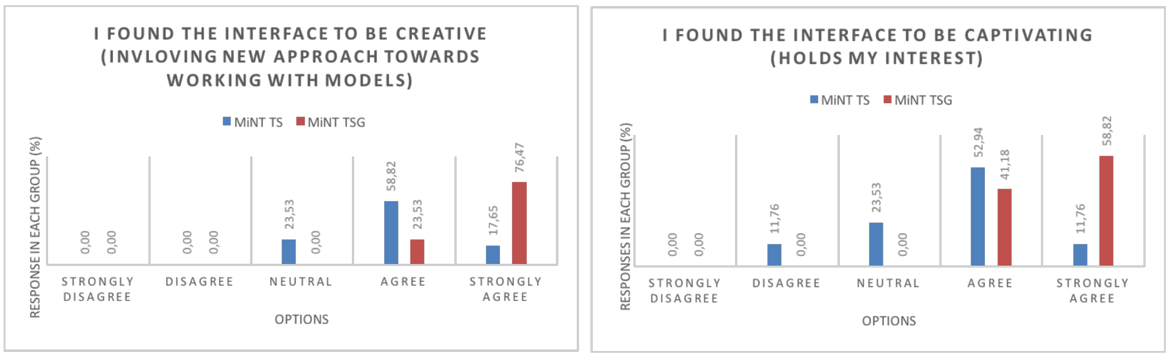


Fig. 7.3 Evaluation of hedonic qualities for touch and speech interface against touch, speech (MiNT TS) and touch gesture interface (MiNT TSG)

of the population (58.8%) decided to opt for neutral or simply disagreed (17.65%) on the statement. Several subjects verbalized the rationale as speech recognition being prone to error, fatigue by speaking a lot, or awkwardness of talking to a speech interface in social setups.

Subjects next evaluated both the interfaces for their hedonic attributes, and the observations are presented in Figure 7.3. A large majority of subjects (76.47%) strongly agreed that MiNT TSG is a creative and new approach towards working with models. Remaining 23.53% also affirmed by agreeing on the statement. For MiNT TS there was a comparatively less strong affirmation from a large majority (58.82%), who agreed to the statement. There was also a small group of a subject (23.53%) who opted for a neutral stance on the point of MiNT TS being creative and new from the modeling perspective. All the subjects found MiNT TSG to be captivating and provided their confirmation by agreeing or strongly agreeing on the point. For MiNT TS a majority of subjects responded by agreeing (52.94%) while remaining subjects opted for either disagrees, neutral or strongly agree. Collected data reveals that on hedonic qualities MiNT TSG provided a sense of positive experience, thus receiving either agree or strongly agree as a response from all the subjects who participated in the experiment.

In Figure 7.4 results for the evaluation of both the interface for their attractiveness is presented. A large group of the subject found MiNT TSG to be very appealing for modeling and model transformation by strongly agreeing (82.35%), while for MiNT TS 64.71% agreed and only 23.53% selected strongly agree as an option. Thus, there was a clear mandate that MiNT TSG is overall very attractive among all the modelers. For the interfaces being pleasant to use again, MiNT TSG was preferred by almost 94% subjects by accumulating agree or strongly agree in response. On the other hand for MiNT TS, responses were split among all five points of the Likert scale, with no clear majority. Still if accumulated, around 47% responded by

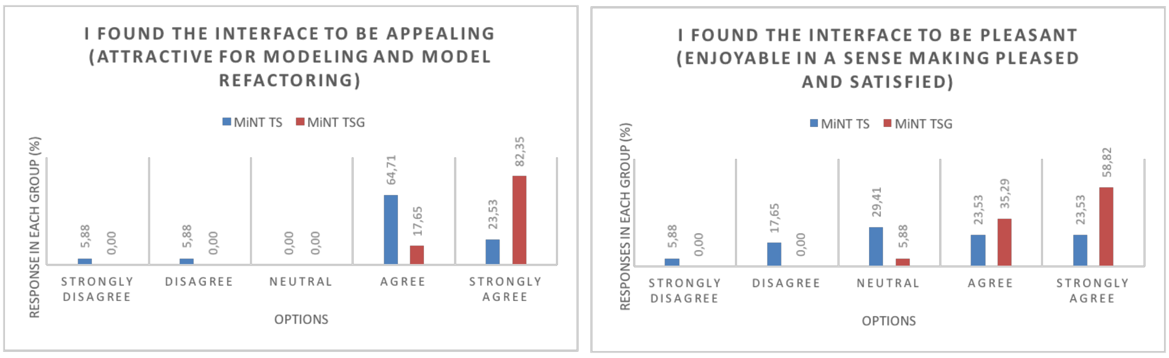


Fig. 7.4 Evaluation of attractiveness qualities for touch and speech interface against touch, speech (MiNT TS) and touch gesture interface (MiNT TSG)

agreeing or strongly agreeing on MiNT TS being enjoyable. One subject (5.88%) responded by taking a neutral stance for MiNT TSG, and further went on to mention again as the steep learning curve being the rationale. We think if given more time to work with MiNT TSG, the subject would have felt more confident.

Traditional modeling CASE-tools with there unimodal interfaces are restrictive in the sense that modelers can use them only in limited spaces (office, workbench, or confined spaces), while mobile devices enable informality and allow users to work in the wider work context. With MiNT TS and MiNT TSG, being available on mobile devices it is interesting to understand if modelers perceive such multimodal interfaces useful or applicable in the context they interact with models or modeling tools. To this end, we asked the modelers to rate both the interfaces on their suitability in the following work context: (a) meetings; (b) office spaces; (c) public spaces; (d) individual modeling sessions; (e) collaborative modeling sessions, where they think can regularly use either of the two interfaces. Figure 7.5 presents the feedback from subjects for which they were asked to select all contexts that apply to a given interface. 58.82% subjects reported that they would readily use MiNT TSG in meetings to quickly realize design changes, while 35.29% subjects considered MiNT TS suitable in such modeling context. Similarly, for office spaces, 64.71% reported finding MiNT TSG applicable, whereas only 35.29% agreed on the same statement for MiNT TS. A few subjects detailed their views as; they would not like to disturb other colleagues by continuously talking with the interface, while with MiNT TSG they believe to be still able to work with less or no speech input if necessary.

None of the subjects considered MiNT TS suitable for public spaces for example during transit using public transportation. For MiNT TSG also subjects were reserved affirming to the statement and only 35.29% responded by agreeing to the statement. We received some explanations covering technical and social concerns such as; internet connection drops in

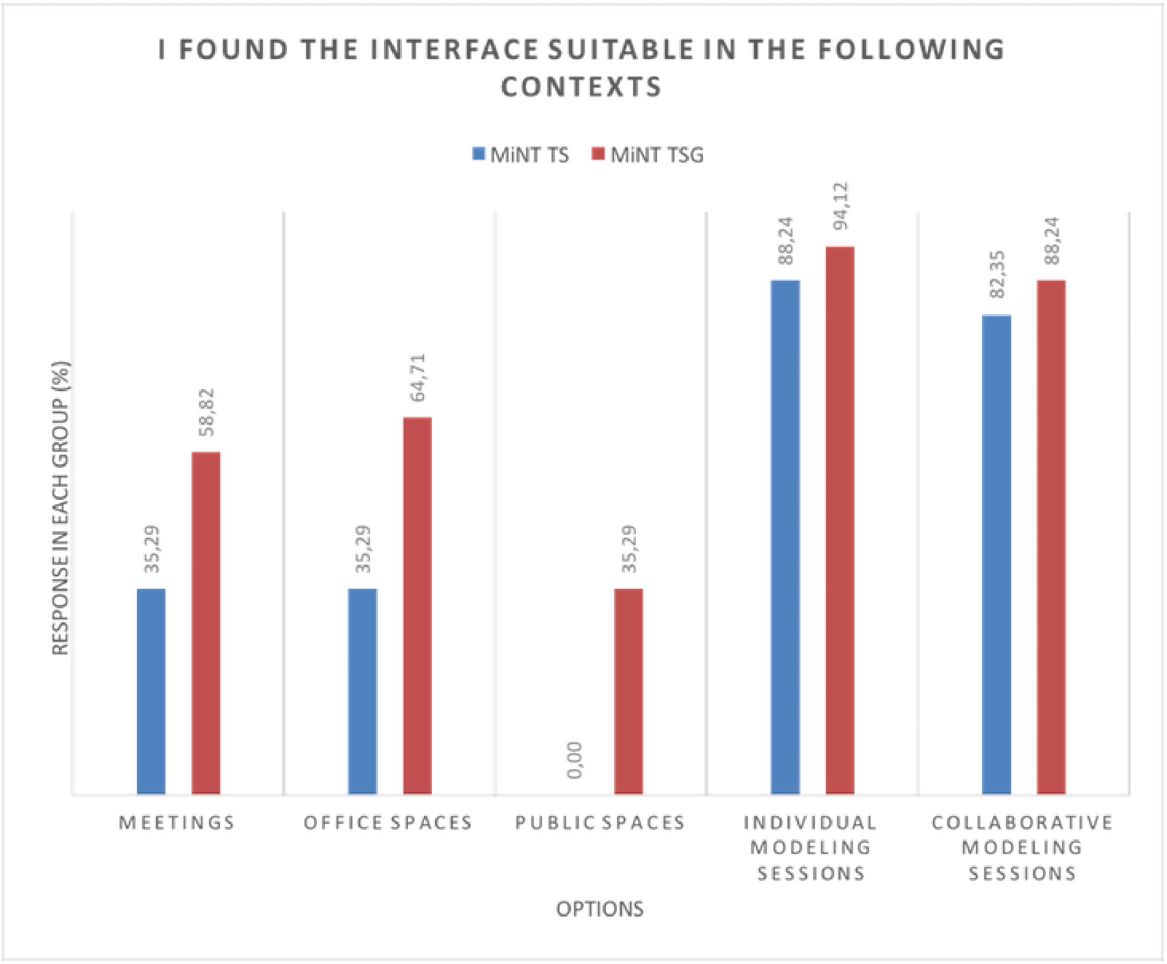


Fig. 7.5 Evaluation of suitable working contexts for touch and speech interface against touch, speech (MiNT TS) and touch gesture interface (MiNT TSG)

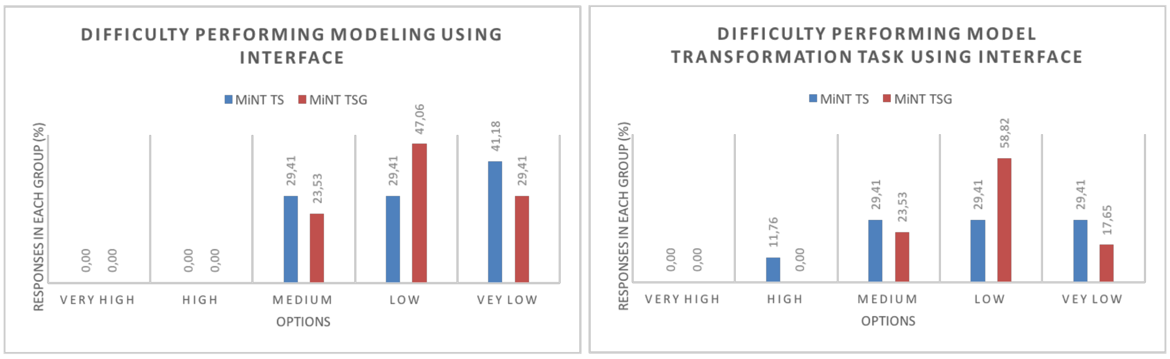


Fig. 7.6 Difficulty performing tasks using touch and speech interface against touch, speech (MiNT TS) and touch gesture interface (MiNT TSG)

transit and speech recognition will not work, strange feeling talking to the interface when others are watching, do not want to disturb fellow passengers, and some even completely rejected the thought of modeling in such a scenario. Next, a large majority of participants agreed that they could use both the interfaces for individual modeling sessions (MiNT TSG=94.12%, MiNT TS=88.24%) and collaborative modeling sessions (MiNT TSG=88.24%, MiNT TS=82.35%) with other modelers and stakeholders. Our observation of the context in which the modelers can use both the interfaces reveals that MiNT TSG has broader applicability over MiNT TS. Although, for both the interfaces participants had their concerns on using speech as an input modality when in public locations for MiNT TSG they considered touch gesture as an alternative for speech.

Next, we asked the subjects to evaluate the tools on a Likert scale to present the difficulty posed by the interface during modeling and model transformation task performed during the experiment (Figure 7.6). Likert scale for the statements presented to the user ranged from difficulty being very high to very low. For modeling task, 41.18% subjects reported that the difficulty with MiNT TS was very low, while with MiNT TSG 29.14% subjects reported the same. A larger group of participants (47.06%) found lower difficulty with MiNT TSG in comparison to MiNT TS interface (29.41%). Modeling was the first task no matter the order of interface to start with for each subject when working with the MiNT TSG. As we already learned from the response of the subject of the steep learning curve with MiNT TSG, we think it to be a reason MiNT TSG did not have a clear mandate of having very low difficulty. On the other hand, subjects while working with MiNT TS interface reported that as the speech recognition works so smooth, they found it extremely supportive to be able to perform the experiment task. For model transformation task, there was an affirmation from a small group for both the interfaces having very low difficulty (MiNT TSG=17.65%, MiNT TS=29.14%), whereas a large group found MiNT TSG (58.82%) to be having low difficulty

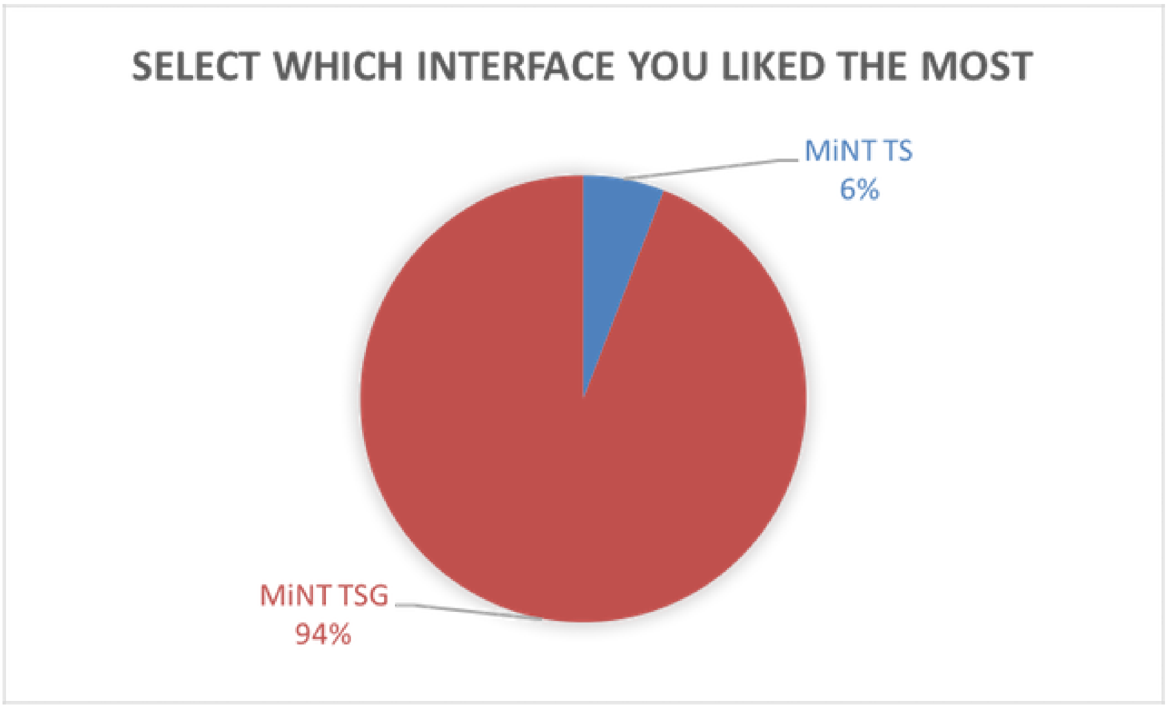


Fig. 7.7 Summative evaluation of touch and speech interface against touch, speech (MiNT TS) and touch gesture interface (MiNT TSG)

over MiNT TS interface (29.41%). For medium difficulty option both the interfaces received relatively fewer responses, and from an almost identical number of subjects. Surprisingly a few subjects reported MiNT TS to be introducing high difficulty when working on model transformation. We could not derive any rationale for their response, even though during the experiment subjects neither faced any speech recognition issues nor the time taken to complete the task was higher than the average. Fatigue caused by speaking continuously to the interface can be a possible reason in this scenario.

Finally, we asked the subjects to rate the interface they liked the most after performing their tasks and would use for daily modeling tasks. Figure 7.7 presents the result in a pie-chart format. 96% clearly opted for MiNT TSG as an interface of their choice, while only 6% (i.e. one subject) selected MiNT TS as the choice of preferred modeling tool interface. This clear mandate from the modelers makes it obvious that a combination of touch, speech, and touch gesture is found to be more useful and practical from a group of experienced modelers.

7.7 Discussion

The multimodal interface developed during this dissertation that combines touch, speech, and touch gesture shows significant improvement in efficiency of modelers for modeling and model transformation process both. Statistical analysis with paired sample t-Test methodology highlights high significance in the data set, and the difference of the mean reports less time taken by modelers if using MiNT TSG, irrespective of the order. Based on our observation, statistical results, and subjects feedback, we found that MiNT TSG requires more input from the subjects to get started with (learning curve) if compared with the interface employing two modalities (MiNT TS with touch and speech). It was found that the 5 minutes time allocated to familiarize the interface was not sufficient to grasp the interplay of three modalities, and hence subjects felt less confident during the modeling task if the initial interface was MiNT TSG.

Subjective evaluation of both the interface on pragmatic, hedonic and attractiveness qualities provide further evidence of MiNT TSG being very useful and practical. While MiNT TSG did not have any significant differentiation from MiNT TS on the ease of use attribute, it was clearly considered *very practical* by a vast majority. On the hedonic qualities, subjects provided clear mandate for MiNT TSG to be more creative and captivating of the two interfaces. Similarly, on attractiveness qualities also subjects found a combination of three modalities far more appealing and pleasant to work with during modeling and model transformation process.

Further subjective evaluation reveals that a combination of touch and speech as input modality is considered suitable for individual modeling sessions or during collaborative modeling sessions. But, introducing one more modality as it is with MiNT TSG makes the interface applicable to broader working contexts. Performing tasks, no matter whether it is modeling or model transformation, MiNT TSG was found to be introducing low difficulty for the modelers and considered ideal for early stage requirements engineering process.

Finally, an overwhelming number of modelers (96%) reported MiNT TSG as the interface they would like to use for their regular modeling and model transformation tasks.

7.8 Threats to Validity

In this section, we discuss how we addressed various threats to the validity of the results.

Since every subject had to perform the same set of tasks using two different, but not the mutually exclusive set of tool specific functionality and interactions, we identify learning curve effect as a major threat to the validity of the results. Since MiNT TS and MiNT TSG both the interface were available as part of the same tool implementation and the two modalities touch and speech are common to both the interfaces, this can affect the efficiency of modelers. Once familiar with the usage of an interface, modelers can get started with the second interface faster. To reduce the effect of learning curve biasing the efficiency, we randomly assigned nine subjects to work with MiNT TS first, and the remaining to work with MiNT TSG first. Our observation shows that both the groups while working with MiNT TSG took less time to work on both the tasks. This randomized assignment also covered the selection bias.

To reduce the experimenter's influence and involvement, task description, existing model for model transformation task, and a list of speech commands, and gesture list was provided to the subjects. Personal interaction was limited to the first phase of the experiment in which experimenter demonstrated how to interact with the interface and clarified any questions or doubts raised by subjects.

Chapter 8

Conclusion and Future Work

Modeling tools are reported to suffer from usability issues and limit the productivity of modelers by hiding functionalities behind complex menus and actions. Making them unusable during the early stages of requirements engineering process when the objective is to build a shared understanding of the system in collaborative sessions involving different stakeholders.

This dissertation employed multimodal interfaces for modeling and model refactoring with the objective to improve the usability of modeling tools, resulting in improved efficiency of the modelers. MiNT framework was designed and developed as a platform for the developers of modeling tools to integrate and benefit from multimodal integration in their modeling CASE-tools. The framework supports the integration of touch gesture, hand and finger motion, interactive whiteboard surfaces, speech input, along with traditional mouse and keyboard input to create natural interfaces.

Two separate reference implementations instantiate the framework to study the feasibility of multimodal interfaces for modeling and model refactoring. Observations from the two controlled experiment using the reference implementation reveal that multimodal interfaces improve the usability as well as makes the modelers more efficient.

This chapter summarizes the contributions (see section 8.1) of this dissertation and highlight the limitations (see section 8.2). In section 8.3 we discuss the future research directions.

8.1 Contributions

In this section we describe the main contributions of this dissertation.

M^3 framework

We presented M^3 framework which unifies the generic multi-modality model with the use case meta-model. M^3 framework allows the designers of multimodal interfaces to capture the modality integration information and associate it with the use case meta-model. This information once established can be linked to other artifacts such as analysis model elements, and for understanding the rationale and evaluating the multimodal interactions on usability goals.

MiNT framework

This dissertation introduces MiNT framework for developers to prototype multimodal modeling interfaces. The framework is highly extensible by design to add new modalities and extend existing modalities such as touch input, gesture input, motion input, speech input, and mouse-keyboard input for multimodal fusion. The framework extends the flexibility of modality integration by allowing the developers to create new unimodal or multimodal interaction definitions in a human-readable markup language format.

Reference implementation for multimodal modeling

The use of MiNT framework is manifested in two reference implementations. MiNT Eclipse enables collaborative face-to-face modeling by employing an interactive whiteboard surface and speech input. MiNT Eclipse also facilitate desktop style modeling session using motion input and speech input modality. MiNT Mobile, the second reference implementation uses touch, speech, and touch gestures on multi-touch tablet devices. By employing three modalities for working with models i t addresses the wider working contexts of the modelers as oppose to MiNT Eclipse which is suitable for office or meeting room scenarios.

Controlled experiments for multimodal modeling and model refactoring

Two controlled experiments were performed to study the feasibility and applicability of multimodal interfaces for modeling and model refactoring. Anecdotal evidence and statistical analysis from the first experiment reveal that employing speech as an input modality *improves the efficiency* of modelers. Participants reported that speech as an input modality enabled them to verbalize their thoughts, and increased the interactivity.

Evidence from the second controlled experiment, statistical observations, and subjective feedback of the modelers support the second hypothesis of this thesis that multimodal interface with a combination of touch, speech, and touch gestures are more *useful* and *practical* as oppose to a multimodal interface employing only touch and speech input. A combination of three modalities improves the efficiency of modelers by allowing them to use more than one modality in parallel and makes the interface suitable for wider modeling context. Subjective evaluation validates the usefulness of the interface employing three modalities.

8.2 Limitations

We identify the participants modeling awareness in both the experiment as the main limitation of this dissertation. The experiment participants had prior modeling expertise and thus do not represent an application domain expert with no modeling expertise. The participants rather fulfilled the requirements a modeler. One of the main objectives of this dissertation was to improve the usability of modeling interfaces such that it encourages the participation of domain experts in collaborative modeling sessions. We think as part of the future work, more thorough investigation and experiments need to be conducted involving application domain experts to study the effectiveness of multimodal interfaces for the mentioned objective.

8.3 Future Work

Through the course of this dissertation, we identified several topics that can be investigated in the future. In the following, we describe possible improvements and different dimensions for extension of the presented work.

Error correction in Multimodal Modeling

Speech recognition error remains a major problem in the design and development of multimodal interfaces with speech as an input modality. Observations made during the controlled experiments indicate that speech recognition errors lead to increased working time and occasionally to the execution of unintended commands, leading to additional user interactions in the form of undo-redo operations. Brinton and colleagues [BFS88] suggested repeating the commands. Murray et al. [MFJ93] raised the issue that repeating might not solve the recognition errors and proposed an elimination-based approach to facilitate correction of misrecognition. Suhm et al. [SMW01] employ a multimodal interface for a dictation task and identify that multimodal speech correction is faster than unimodal correction by repeating the spoken command. They improve the correction accuracy by employing algorithms that use context information for error correction.

MiNT provides a limited amount of error correction based on the work from Suhm and his colleagues by allowing a user to switch from one modality to another to correct an erroneous input. However, error prevention and correction during multimodal modeling remains a topic for future work.

Collaborative Multimodal Modeling

Collaboration is an inseparable part of the modeling, a knowledge acquisition, and building process. Within the scope of this dissertation, we identified that collaborative modeling could benefit from seamless integration of multimodal interfaces, for example during early stages of requirement analysis domain experts can provide input using speech as an input modality. Or a group of modelers works on the same model in a real-time collaborative environment using multimodal interface employing speech and gestures. More work needs to be done in future to study the collaboration of the domain experts and modelers using multimodal interfaces.

Large-scale Study of Multimodal Acceptance

This dissertation provides evidence for the applicability and acceptance of multimodal interfaces by modelers. In the future, it will be interesting to deploy the multimodal interfaces for modeling to the general everyday modelers and study the usage. It will allow gaining more insight into the acceptance of multimodal modeling interfaces by a larger modeling population and the collecting user feedback for possible improvements.

Output Modality Enablers for Virtual and Augmented Reality

This dissertation primarily focused on the input modalities for multimodal fusion. Technological limitations have traditionally restricted modeling within the two-dimensional representation of computer screens enabling visual output coupled with auditory modality. Advancement and wider availability of virtual reality and augmented reality device open a new paradigm for experimenting with how we visualize and conceptualize models. Can augmenting real-world objects by overlaying the model elements on the top enhance model comprehension, or can virtual reality provide more freedom to interact with models in room scale visualization? These ideas are worth investigation and a possible future towards solving complex problems collaboratively.

Appendix A

Model Refactoring Taxonomy

In this section, we detail some of the most common model refactoring use case that is considered throughout this thesis.

1. **Pull Up Method:** Duplicate methods in the subclasses can be a source for error and unnecessary complexity. The most common scenario for this refactoring task is the scenario when two methods have the similar body, attributed to code duplicates. If the methods are supposed to have an identical purpose, duplication can result in overhead in keeping both of them updated in case of any changes.

2. **Pull Up Attribute:** Duplicate attributes can be a source of a bug. If the subclasses have attributes serving the similar purpose, it is advisable to benefit from the generalization concept by moving the attribute declaration to the superclass.

3. **Extract Superclass:** Duplicate functionality in the code is a common is a principal cause associated with bad design. It can further increase the overhead of keeping the similar functionality updated in the classes as well as separately testing them for any regression fault. The idea behind this refactoring task is to benefit from inheritance concept of object oriented programming. An alternative to Extract Superclass refactoring task is Extract class. Modeler has a choice between inheritance and delegation. Inheritance is a simpler choice if the two classes share similar interface and behavior.

4. **Push Down Attribute:** This refactoring task is opposite of Pull Up Attribute. This is performed by moving an attribute from a superclass to a subclass if the attribute is specific to the subclass.

5. **Push Down Method:** This task is opposite of Pull Up Method. This is performed by moving a method from a superclass to a subclass if the behavior is specific to the subclass.

6. **Extract Subclass:** Input to this refactoring task is a class with features which are only used by some instances of the class or a class with features accumulated over time (e.g. Blob anti-pattern). An alternative to Extract subclass refactoring task is Extract class. Again the choice is between inheritance and delegation. Extract subclass is simpler over delegation but restricts the class-based behavior of an object as a subclass represents a set of variations in the hierarchy. On the other hand, Extract class allows having behavior which is not necessarily highly coupled.

7. **Collapse Hierarchy:** Too detailed inheritance hierarchy is the prime target for this refactoring task. After moving methods and fields in the hierarchy tree, subclasses can become obsolete and necessarily not be adding any value. In such a scenario this refactoring task is performed to merge the classes together and reduce the hierarchal depth.

8. **Merge Classes:** Closely associated behavior are advisable to be kept together for simplicity and maintenance purposes. This refactoring task aims at achieving that by merging classes with similar behavior by moving fields and attributes from one class to another and removing the obsolete class.

9. **Replace Data Value with Object:** Frequently in early stages of development basic data types are used to represent attributes in a class to capture simple facts. But as the development proceeds, these simple attributes can require having additional attributes and behavior leading them to look like an independent object in itself. For example, a user name represented as a string can at a later stage require more information like user address, or user phone number, which could be grouped together. This refactoring task assists in dealing with such scenario by creating a new class and moving the simple attribute from the source class to the new class. Additional attributes and behaviors can further be added to the new class. Finally, a reference to this new class is left in the source class.

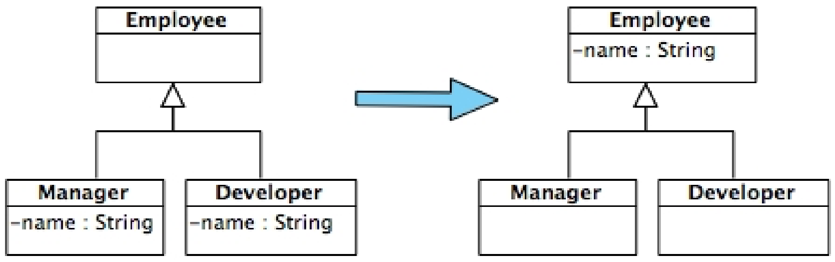


Fig. A.1 Pull up attribute use case

Use case	**PullUpAttribute**
Participating actors	Instantiated by *Modeler*
Flow of events	1. ***UserStep:*** *Modeler* selects two or more classes and executes **Pull up attribute** use case 2. ***SystemStep:*** *CASE-tool* moves all the common attributes of the selected classes to the superclass 3. ***SystemStep:*** *CASE-tool* notifies user of the successful execution of the use case
Entry condition	• *Modeler* has selected two or more classes for the use case
Exit condition	• *CASE-tool* presents user a superclass with all the common attributes of the selected subclasses

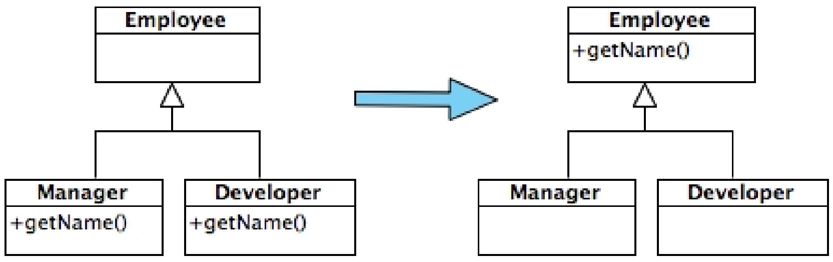


Fig. A.2 Pull up method use case

Use case	**Pull up method**
Participating actors	Instantiated by *Modeler*
Flow of events	1. ***UserStep:****Modeler* selects a class and executes **Pull up method** use case 2. ***SystemStep:*** *CASE-tool* moves all the common methods of the selected class to the superclass 3. ***SystemStep:*** *CASE-tool* notifies user of the successful execution of the use case
Entry condition	• *Modeler* has selected two or more classes for the use case
Exit condition	• *CASE-tool* presents user a superclass with all the common methods of the selected subclasses

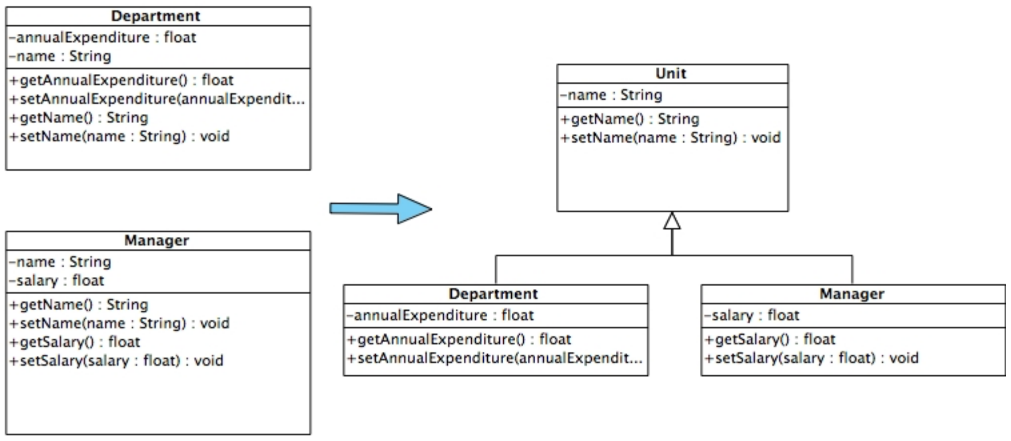

Department
-annualExpenditure : float
-name : String
+getAnnualExpenditure() : float
+setAnnualExpenditure(annualExpendit...
+getName() : String
+setName(name : String) : void
Manager
-name : String
-salary : float
+getName() : String
+setName(name : String) : void
+getSalary() : float
+setSalary(salary : float) : void
Unit
-name : String
+getName() : String
+setName(name : String) : void
Department
-annualExpenditure : float
+getAnnualExpenditure() : float
+setAnnualExpenditure(annualExpendit...
Manager
-salary : float
+getSalary() : float
+setSalary(salary : float) : void

Fig. A.3 Extract superclass use case

Use case	**Extract superclass**
Participating actors	Instantiated by *Modeler*
Flow of events	1. ***UserStep:*** *Modeler* selects two or more classes and executes *Extract superclass* use case 2. ***SystemStep:*** *CASE-tool* creates a new abstract superclass, and makes the *Modeler* selected classes its subclass 3. ***SystemStep:*** *CASE-tool* executes **Pull up attribute**, **Pull up method** use cases to move common elements to the superclass 4. ***SystemStep:*** *CASE-tool* notifies user of the successful execution of the use case
Entry condition	• *Modeler* has selected more then one classes for the use case
Exit condition	• *CASE-tool* presents user with a superclass of the selected classes that contains all the common attributes, and methods of the subclasses

Use case	**Push down attribute**
Participating actors	Instantiated by *Modeler*
Flow of events	1. ***UserStep:*** *Modeler* selects an existing superclass along with zero or more subclasses, and executes *Push down attribute* use case for every desired attribute 2. ***SystemStep:*** *CASE-tool* moves the selected attribute to the user selected subclass. If user did not select a subclass, attribute is moved to all the existing subclasses
Entry condition	• *Modeler* has selected at least one class for the use case
Exit condition	• *CASE-tool* presents user with desired attribute moved to the subclass

Use case	**Push down method**
Participating actors	Instantiated by *Modeler*
Flow of events	1. ***UserStep:*** *Modeler* selects an existing superclass along with zero or more subclasses, and executes **Push down method** use case for every desired method 2. ***SystemStep:*** *CASE-tool* moves the selected method to the user selected subclass. If user did not select a subclass, method is moved to all the existing subclasses
Entry condition	• *Modeler* has selected at least one class for the use case
Exit condition	• *CASE-tool* presents user with desired method moved to the subclass

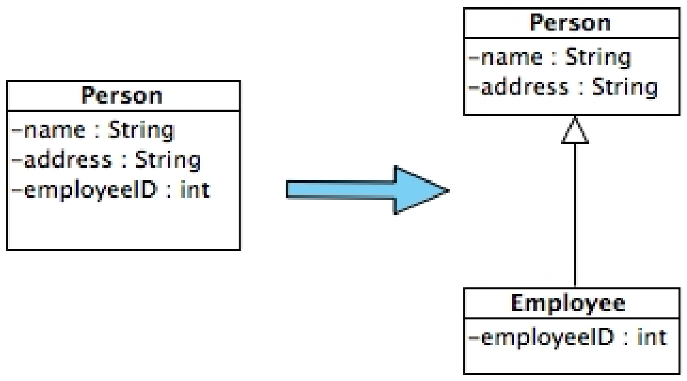


Fig. A.4 Extract subclass use case

Use case	**Extract subclass**
Participating actors	Instantiated by *Modeler*
Flow of events	1. ***UserStep:*** *Modeler* selects an existing class and executes **Extract subclass** use case 2. ***SystemStep:*** *CASE-tool* presents user a new class with generalization relationship to the selected class 3. ***UserStep (optional):*** *Modeler* executes **push down attribute**, **push down method** use cases to move elements from superclass to the subclass
Entry condition	• *Modeler* has selected one class for the use case
Exit condition	• *CASE-tool* presents user with a subclass of an existing class

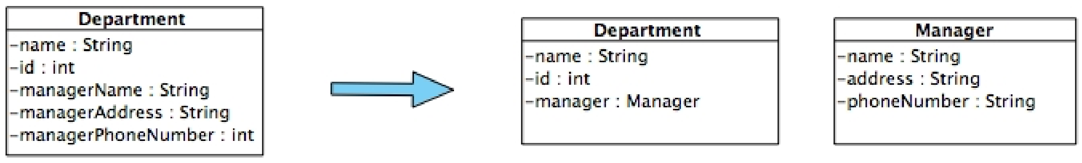


Fig. A.5 Attribute to object use case

Use case	**Attribute to object**
Participating actors	Instantiated by *Modeler*
Flow of events	1. ***UserStep:*** *Modeler* selects one of more attributes from an existing class and executes **Attribute to object** use case 2. ***SystemStep:*** *CASE-tool* moves attributes and the getter/setter of the attributes to the new class 3. ***SystemStep:*** *CASE-tool* changes the type of the attribute in the source class to the new class
Entry condition	• *Modeler* has selected attributes from an existing class
Exit condition	• *CASE-tool* presents user with a new class with existing attributes and a reference variable in the source class

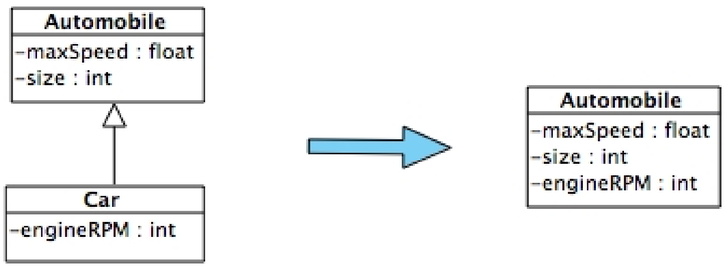


Fig. A.6 Collapse hierarchy use case

Use case	**Collapse hierarchy**
Participating actors	Instantiated by *Modeler*
Flow of events	1. ***UserStep:*** *Modeler* selects an existing superclass along with zero or more subclasses to execute Collapse hierarchy use case 2. ***SystemStep:*** *CASE-tool* moves all attributes and methods from the selected subclass and moves them to superclass. If no subclass was selected method and attributes from all immediate subclasses are moved to the superclass 3. ***SystemStep:*** *CASE-tool* removes the subclass/es 4. ***SystemStep:*** *CASE-tool* notifies user of the successful execution of the use case
Entry condition	• *Modeler* has selected a superclass
Exit condition	• *CASE-tool* presents user with a superclass which contains all the attributes and methods of the subclass/es

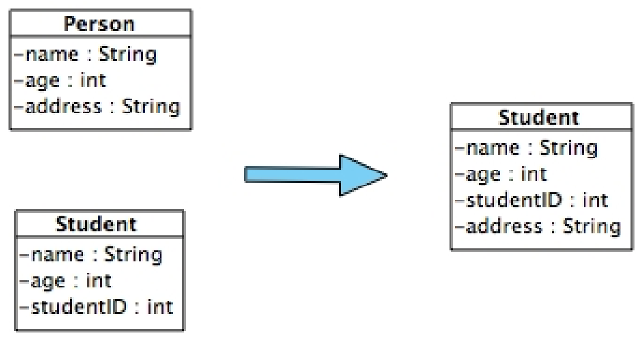


Fig. A.7 Merge classes use case

Use case	**Merge classes**
Participating actors	Instantiated by *Modeler*
Flow of events	1. ***UserStep:*** *Modeler* selects source and target class and executes **Merge classes** use case 2. ***SystemStep:*** Attributes and methods from source class is moved to the target class 3. ***SystemStep (optional):*** Common attributes and methods in source and target class is moved to superclass by **extract superclass** use case 4. ***SystemStep:*** Source class is removed from the model
Entry condition	• *Modeler* has selected a source and target class
Exit condition	• Source class attributes and methods are either moved to target class or the superclass of the target class

Appendix B

Controlled Experiment

B.1 Instructors Checklist

Part 1 : Before the experiment

1. Introduce participant selected for multimodal interface treatment on how to work with the interface. Provide a list of available speech commands.
2. Provide basic tool usage information to the participant selected for baseline treatment (only if they request for it).
3. Answer any questions participant might have on the tool usage.
4. Notify participant that the modeling session will be video recorded for further analysis. It will only be viewed only for research purposes by the investigator and subsequently removed from every storage once they are transcribed or coded. Every participant's participation is treated as ***anonymous*** within the course of this experiment and an appropriate level of confidentiality will be maintained about the participation. If you do not wish to be recorded, please notify the instructor immediately.
5. Notify participant that they have to employ **think a loud** methodology while working on the modeling tasks.
6. Inform the participant that they have to perform two modeling tasks. Task description for the next task will be provided when the participant notifies the instructor of the completion of the current task. Once the task description is provided, the participant should start with the modeling task as soon as possible, and also the participant should notify instructor when they are ready to model after reading the task description.
7. Start recording session.
8. Distribute first task description.

Part 2: During the experiment

1. Monitor participants behavior and make notes of any subjective observations.
2. Pause recording session if the participant notifies of a task completion.
3. Hand in the subsequent task description to the participant as soon as he is ready, and also resume the recording session.

Part 3 : After the experiment

1. Stop video recording.
2. Distribute questionnaire to the participant and ask them to fill it.
3. After they have finished with the questionnaire, discuss with them their rationale behind each of the answers. Note down any observations that might be interesting. Participant can give their subjective perception on the usability of the multimodal interface, and if any observation they want to share with the instructors.
4. Thanks the participant for their participation, and ask them to leave their email address if they would like to have a copy of the video recording.
5. Store artifacts from each participation in a labeled set.

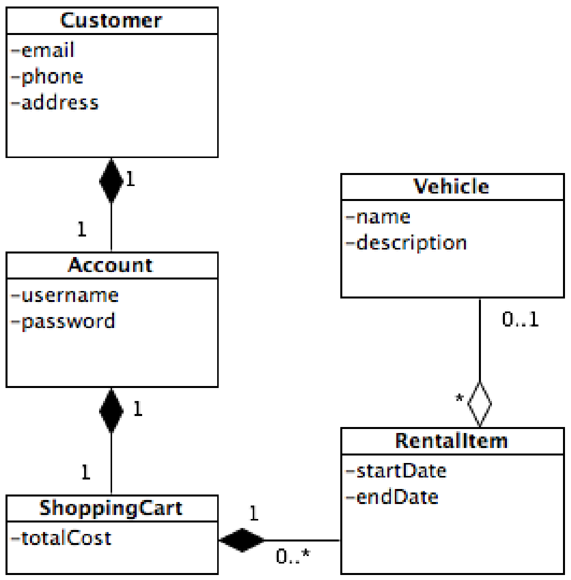


Fig. B.1 Sample solution object model for Task 1

B.2 Experiment Task

Task 1	**Modeling : Online Rental Platform**
Context	You are developing an online platform for vehicle rental. You are required to model the basic domain objects, their attributes and how they associate with each other (**ignore methods, types, and multiplicity for this task**).
Description	The **Online Rental Platform** allows a *Customer* to rent the *Vehicle*. A customer needs to have an *Account* on the system to rent a vehicle. Every customer should provide their *email*, *phone* and *address* information. An account must have a *username*, and *password* of the customer. A customer can select vehicle by providing *start* and *end* date, and add them to their *Shopping cart*. Shopping cart stores all the *Rental items*, and shows the *total cost* to the customer for all the vehicles. Each vehicle has a different *name* and a short *description*.

Task 2	**Model transformation (Refactoring and Refinement) : Rental Company**
Context	The provided model represents a rental company (see figure B.2).
Description	Improve the existing model using model refactoring and refinement based on the transformation steps given below. In a meeting, the following model improvements have been decided. You as a modeling expert are asked to refactor the model without changing the underlying functionality. Do the following model transformation on the provided model: • The distinction between the *AdministrativeStaff* and *TechnicalStaff* is no longer necessary. Except for the *joiningDate* attribute, which should be moved to the *Support* class. • *Manager* and *Support* both should be modeled with an *Employee* class and associated with the *Department* class. Common attributes and associated getters should be moved to the *Employee* super class. • The *Contact* class has two attributes *customerId* and *supplierId*. Move these attributes to their respective classes, create new classes if needed. • The *Customer* and *Lawyer* class contain duplicate methods and attributes that should be moved to *Contact* class. • Prepare *RentalCompany* and *Contact* for extensibility using delegation. Turn the address attribute into a separate class and use delegation to access it from the *RentalCompany* and *Contact*. The *Address* should also contain two new attributes country *name* and *postal code*.

B.3 Subjective Questionnaire

Date:
Participant ID:

1. Select your level of domain expertise for each of the tasks in the experiment.

   - Task 1: Online Rental Platform Low ☐ Medium ☐ High ☐
   - Task 2: Organizational Hierarchy Low ☐ Medium ☐ High ☐

2. Select how frequently you are using modeling tools and techniques.

   Never ☐ Once-twice per month ☐ Once-twice per week ☐ Everyday☐

3. Which task was most difficult?

   Task 1 ☐ Task 2 ☐

4. Was the provided material (task description and model) difficult to understand for the most difficult task?

   Yes ☐ No ☐

5. Did you create some kind of to-do list (external or in your mind) after reading the task description or after looking at the model for each of the tasks?

   Yes ☐ No ☐

6. **If answered 'YES' to question 5:** Did you think about required actions (tool-specific) while creating the to-do list before starting to work on the task, or you thought about tool-specific actions during the task?

   Yes, I thought **before** starting to work ☐ Yes, I thought **during** the task ☐

7. Rate the difficulty of performing each task using the modeling tool interface.

   - Task 1: Very Low ☐ ☐ ☐ ☐ ☐ Very High
   - Task 2: Very Low ☐ ☐ ☐ ☐ ☐ Very High

If you have additional comments you can add them here:

Thank you for your participation.
Please leave your email address if you want to receive a copy of the results.

Email:

B.4 Stastical Analysis: F-test for variance equality

F-Test Two-Sample for Variances		
	Group Baseline (Task 1)	Group M3 (Task 1, no overhead)
Mean	474,6666667	348,2857143
Variance	40547,86667	3548,571429
Observations	6	7
df	5	6
F	11,42653247	
P(F<=f) one-tail	0,00504258	
F Critical one-tail	4,387374187	

F-Test Two-Sample for Variances		
	Group Baseline (Task 1)	Group M3 (Task 1, speech overhead)
Mean	474,6666667	424
Variance	40547,86667	3817,333333
Observations	6	7
df	5	6
F	10,62203982	
P(F<=f) one-tail	0,006095738	
F Critical one-tail	4,387374187	

F-Test Two-Sample for Variances		
	Group Baseline (Task 2)	Group M3 (Task 2, no overhead)
Mean	588,8333333	395,5714286
Variance	31842,96667	5134,285714
Observations	6	7
df	5	6
F	6,202024671	
P(F<=f) one-tail	0,023031146	
F Critical one-tail	4,387374187	

F-Test Two-Sample for Variances		
	Group Baseline (Task 2)	Group M3 (Task 2, speech overhead)
Mean	588,8333333	455,7142857
Variance	31842,96667	6051,571429
Observations	6	7
df	5	6
F	5,261933539	
P(F<=f) one-tail	0,033581736	
F Critical one-tail	4,387374187	

B.5 Stastical Analysis: t-Test for unequal variances

t-Test: Two-Sample Assuming Unequal Variances		
	Group Baseline (Task 1)	Group M3 (Task 1, no overhead)
Mean	474,6666667	348,2857143
Variance	40547,86667	3548,571429
Observations	6	7
Hypothesized Mean Difference	0	
df	6	
t Stat	1,482744455	
P(T<=t) one-tail	0,094330822	
t Critical one-tail	1,943180281	
P(T<=t) two-tail	0,188661644	
t Critical two-tail	2,446911851	

t-Test: Two-Sample Assuming Unequal Variances		
	Group Baseline (Task 1)	Group M3 (Task 1, speech overhead)
Mean	474,6666667	424
Variance	40547,86667	3817,333333
Observations	6	7
Hypothesized Mean Difference	0	
df	6	
t Stat	0,592874037	
P(T<=t) one-tail	0,287457532	
t Critical one-tail	1,943180281	
P(T<=t) two-tail	0,574915064	
t Critical two-tail	2,446911851	

t-Test: Two-Sample Assuming Unequal Variances		
	Group Baseline (Task 2)	Group M3 (Task 2, no overhead)
Mean	588,8333333	395,5714286
Variance	31842,96667	5134,285714
Observations	6	7
Hypothesized Mean Difference	0	
df	6	
t Stat	2,486595366	
P(T<=t) one-tail	0,023690076	
t Critical one-tail	1,943180281	
P(T<=t) two-tail	0,047380152	
t Critical two-tail	2,446911851	

t-Test: Two-Sample Assuming Unequal Variances		
	Group Baseline (Task 2)	Group M3 (Task 2, speech overhead)
Mean	588,8333333	455,7142857
Variance	31842,96667	6051,571429
Observations	6	7
Hypothesized Mean Difference	0	
df	7	
t Stat	1,694489209	
P(T<=t) one-tail	0,066998912	
t Critical one-tail	1,894578605	
P(T<=t) two-tail	0,133997823	
t Critical two-tail	2,364624252	

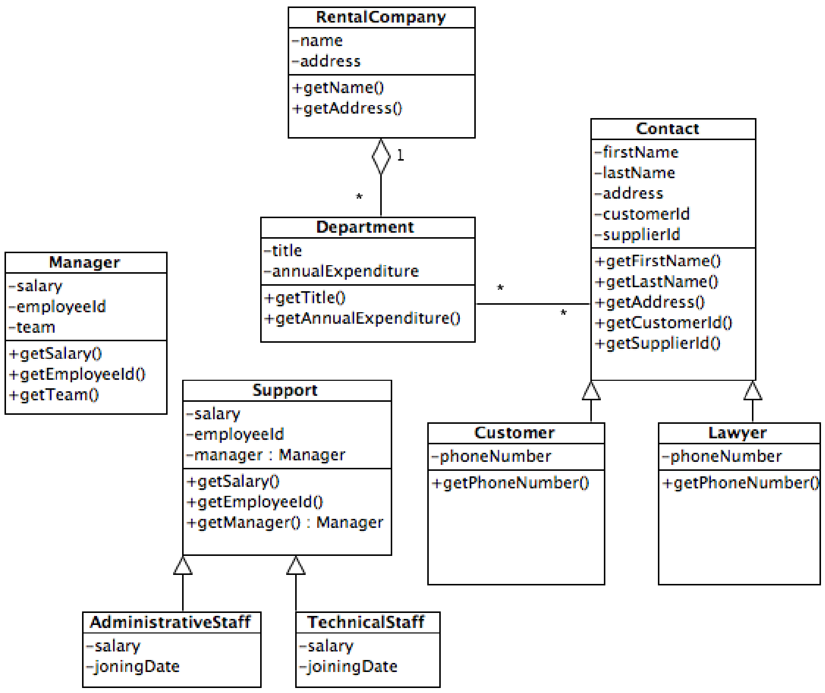


Fig. B.2 Existing object model for Task 2

Appendix C

Controlled Experiment

C.1 Instructors Checklist

Part 1: Before the experiment

1. Introduce participant that they will be performing two tasks using two different interfaces.
2. Interface that should be used first in the experiment will be decided at random.
3. Participant will be introduced to the selected interface and allowed to familiarize themselves with using the interface for five minutes.
4. Once the participant has completed both the task with the first interface, the second interface will be introduced. Participant will again have the possibility to familiarize them with using the interface for five minutes.
5. Once the tasks are completed with an interface, the participant will have to fill the corresponding questionnaire.
6. Notify participant that they have to employ **think a loud** methodology while working on the modeling tasks.
7. Provide first task description.

Part 2: During the experiment

1. Monitor participants behavior and make notes of any subjective observations.
2. Hand over second task description if participant notifies of first task completion.
3. Ask participant to fill the questionnaire for the interface used for the last task.
4. Introduce participant second interface.

Part 3: After the experiment

1. Ask participant to fill the questionnaire for the second interface
2. Thanks the participant for their participation.
3. Store artifacts from each participation in a labeled set.

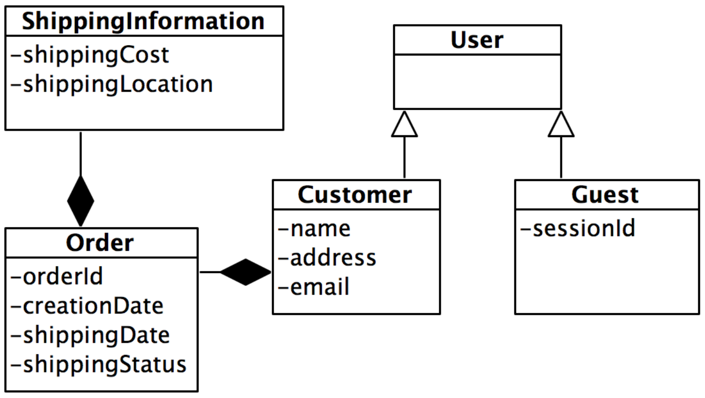


Fig. C.1 Sample solution object model for Task 1

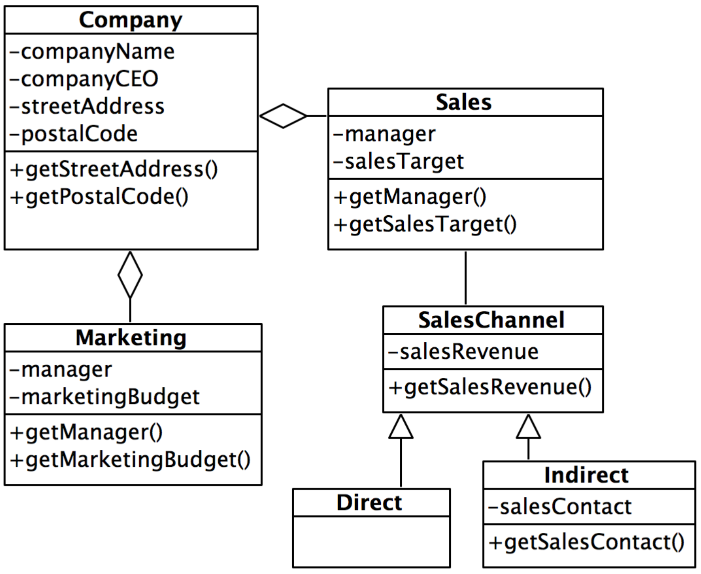


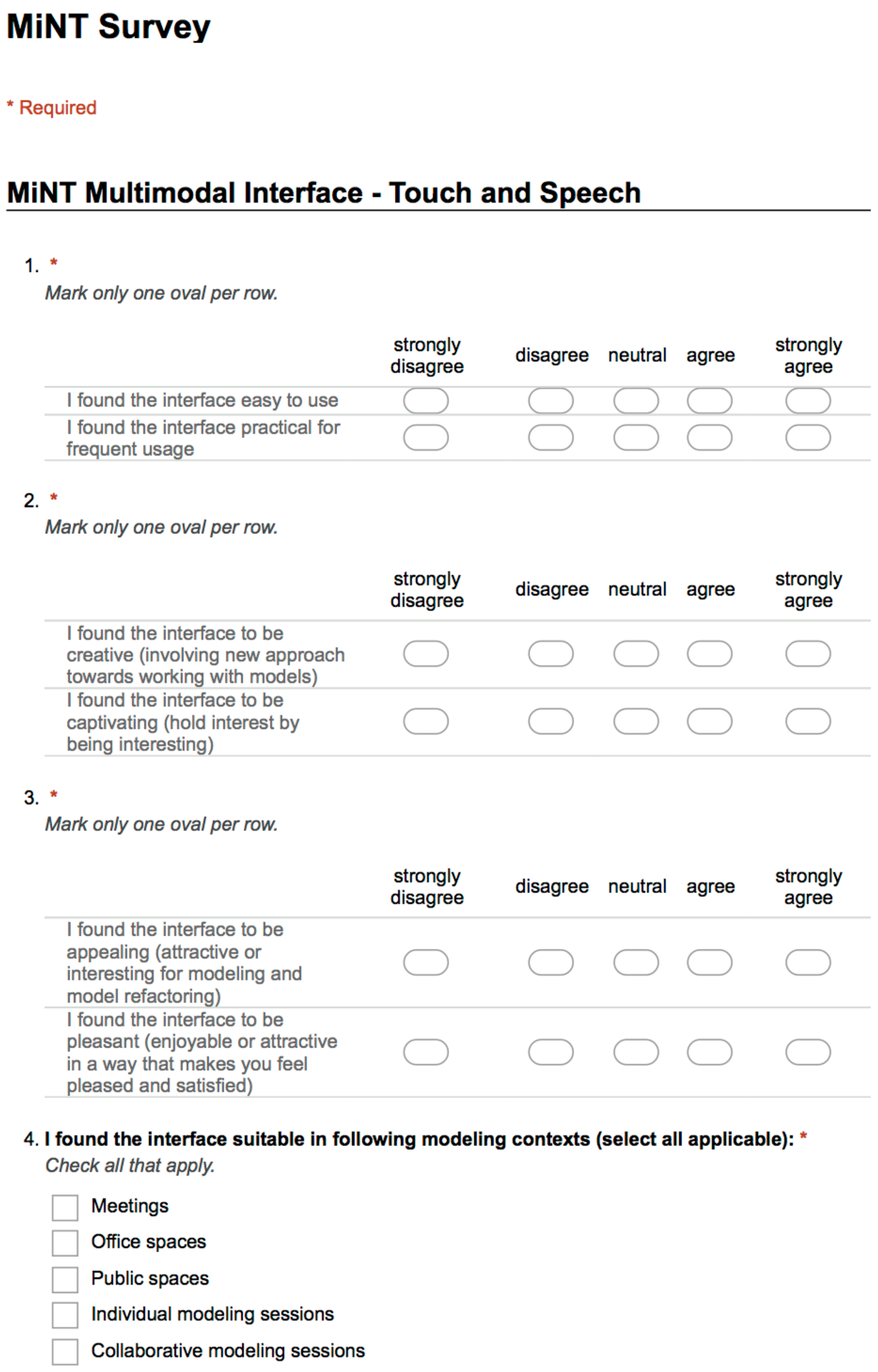
Fig. C.2 Object model for Task 2

C.2 Experiment Task

Task 1	**Modeling : Order tracking**
Context	Your company is developing an online e-commerce platform. You are assigned the task of modeling domain objects, their attributes and how they associate with each other for the order tracking subsystem (**ignore methods, attribute types, method return types, and multiplicity for this task**).
Description	An order tracking subsystem should allow the management of the orders and report their status to the user. This system should support two different types of users; customer, and guest. A *Customer* is a registered user of the system and has provided the *name*, *address*, and *email*. A *Guest* is a user who has not registered with the system. *Guests* have a *sessionId* attribute that is used to keep track of their shopping activity. *Guest* users are required to register as a *Customer* before they can place their *Order*. Every *Order* belongs to a *Customer*. An *Order* has an *orderId*, *creationDate*, *shippingDate* and a *shippingStatus* attribute. Furthermore, every *Order* is associated with a *Shipping Information* that keeps track of *shippingCost*, and *shippingLocation*.

Task 2	**Model transformation (Refactoring and Refinement) : Organizational Hierarchy**
Context	The provided model represents a generic Organizational hierarchy (see figure C.2).
Description	Improve the provided model using model refactoring and refinement based on the transformation steps given below. Do the following model transformation on the provided model: • Extract a super-class *Department* for *Marketing*, and *Sales* classes. Add attribute *name* and *annualBudget* to the *Department* class. • Pull up common attributes and methods from *Marketing* and *Sales* class to the *Department* super class. • Merge *Marketing* and *Sales* as *MarketingAndSales*. • Collapse Hierarchy of the *SalesChannel* class. Attributes and methods from subclasses should be moved to the super class. • Create a class *Address* and use delegation to access it from the *Company* class. Move attributes *streetAddress*, *postalCode* and methods *getStreetAddress* and *getPostalCode* from *Company* to the new class.

C.3 Subjective Questionnaire



MiNT Survey

* Required

MiNT Multimodal Interface - Touch and Speech

1. *

Mark only one oval per row.

	strongly disagree	disagree	neutral	agree	strongly agree
I found the interface easy to use	◯	◯	◯	◯	◯
I found the interface practical for frequent usage	◯	◯	◯	◯	◯

2. *

Mark only one oval per row.

	strongly disagree	disagree	neutral	agree	strongly agree
I found the interface to be creative (involving new approach towards working with models)	◯	◯	◯	◯	◯
I found the interface to be captivating (hold interest by being interesting)	◯	◯	◯	◯	◯

3. *

Mark only one oval per row.

	strongly disagree	disagree	neutral	agree	strongly agree
I found the interface to be appealing (attractive or interesting for modeling and model refactoring)	◯	◯	◯	◯	◯
I found the interface to be pleasant (enjoyable or attractive in a way that makes you feel pleased and satisfied)	◯	◯	◯	◯	◯

4. **I found the interface suitable in following modeling contexts (select all applicable):** *

Check all that apply.

- ☐ Meetings
- ☐ Office spaces
- ☐ Public spaces
- ☐ Individual modeling sessions
- ☐ Collaborative modeling sessions

5. **Difficulty performing tasks using touch and speech interface** *
Mark only one oval per row.

	Very Low	Low	Medium	High	Very High
Task 1 - Modeling : Order tracking	⬭	⬭	⬭	⬭	⬭
Task 2 - Model transformation : Online Banking	⬭	⬭	⬭	⬭	⬭

MiNT Multimodal Interface - Touch, Speech and Gesture

6. *
Mark only one oval per row.

	strongly disagree	disagree	neutral	agree	strongly agree
I found the interface easy to use	⬭	⬭	⬭	⬭	⬭
I found the interface practical for frequent usage	⬭	⬭	⬭	⬭	⬭

7. *
Mark only one oval per row.

	strongly disagree	disagree	neutral	agree	strongly agree
I found the interface to be creative (involving new approach towards working with models)	⬭	⬭	⬭	⬭	⬭
I found the interface to be captivating (hold interest by being interesting)	⬭	⬭	⬭	⬭	⬭

8. *
Mark only one oval per row.

	strongly disagree	disagree	neutral	agree	strongly agree
I found the interface to be appealing (attractive or interesting for modeling and model refactoring)	⬭	⬭	⬭	⬭	⬭
I found the interface to be pleasant (enjoyable or attractive in a way that makes you feel pleased and satisfied)	⬭	⬭	⬭	⬭	⬭

9. **I found the interface suitable in following modeling contexts (select all applicable):** *
Check all that apply.

- ☐ Meetings
- ☐ Office spaces
- ☐ Public spaces
- ☐ Individual modeling sessions
- ☐ Collaborative modeling sessions

10. **Difficulty performing tasks using touch, speech and gesture interface** *

Mark only one oval per row.

	Very Low	Low	Medium	High	Very High
Task 1 - Modeling : Order tracking	⬭	⬭	⬭	⬭	⬭
Task 2 - Model transformation : Online Banking	⬭	⬭	⬭	⬭	⬭

General Information and Feedback

Please provide some general information about yourself and additional feedback or comments with respect to MiNT usage.

11. **Rank which interface you liked the most** *

Mark only one oval.

- ⬭ MiNT Interface - Touch and Speech
- ⬭ MiNT Interface - Touch, Speech and Gesture

12. **Educational qualification (select one that you are pursuing currently or the last one in case not studying anymore)** *

Mark only one oval.

- ⬭ Bachelor
- ⬭ Master
- ⬭ PhD

13. **Experience with modeling tools and techniques** *

Mark only one oval.

- ⬭ less than a year
- ⬭ 1 to 2 years
- ⬭ 2 to 3 years
- ⬭ more than 3 years

14. **Provide feedback, comments, or suggestions to improve MiNT**

15. **Provide your email address if you would like to receive a copy of the result from this survey**

C.4 Statistical Analysis: t-Test paired sample

t-Test: Paired Two Sample for Means - Modeling		
	TS	TSG
Mean	211,2941176	160,5882353
Variance	2129,095588	1972,882353
Observations	17	17
Pearson Correlation	0,504026874	
Hypothesized Mean Difference	0	
df	16	
t Stat	4,633370777	
P(T<=t) one-tail	0,000138071	
t Critical one-tail	1,745883676	
P(T<=t) two-tail	0,000276142	
t Critical two-tail	2,119905299	

t-Test: Paired Two Sample for Means - Model Transformation		
	TS	TSG
Mean	192,9411765	112,8823529
Variance	1046,183824	989,2352941
Observations	17	17
Pearson Correlation	-0,091117505	
Hypothesized Mean Difference	0	
df	16	
t Stat	7,004515124	
P(T<=t) one-tail	0,0000014861	
t Critical one-tail	1,745883676	
P(T<=t) two-tail	0,0000029723	
t Critical two-tail	2,119905299	

t-Test: Paired Two Sample for Means - Modeling - Initial MiNT TSG		
	TS	TSG
Mean	192,375	180,875
Variance	1692,553571	1482,982143
Observations	8	8

Pearson Correlation	0,83879654	
Hypothesized Mean Difference	0	
df	7	
t Stat	1,429544765	
P(T<=t) one-tail	0,097960781	
t Critical one-tail	1,894578605	
P(T<=t) two-tail	0,195921561	
t Critical two-tail	2,364624252	

t-Test: Paired Two Sample for Means - Model transformation - Initial MiNT TSG		
	TS	TSG
Mean	188	135
Variance	1125,142857	504,5714286
Observations	8	8
Pearson Correlation	-0,300894137	
Hypothesized Mean Difference	0	
df	7	
t Stat	3,284440027	
P(T<=t) one-tail	0,006702397	
t Critical one-tail	1,894578605	
P(T<=t) two-tail	0,013404793	
t Critical two-tail	2,364624252	

t-Test: Paired Two Sample for Means - Modeling - Initial MiNT TS		
	TS	TSG
Mean	228,1111111	142,5555556
Variance	2101,111111	1870,777778
Observations	9	9
Pearson Correlation	0,82123406	
Hypothesized Mean Difference	0	
df	8	
t Stat	9,595248381	
P(T<=t) one-tail	0,0000057727	

t Critical one-tail	1,859548038	
P(T<=t) two-tail	0,0000115453	
t Critical two-tail	2,306004135	

t-Test: Paired Two Sample for Means - Model transformation - Initial MiNT TS		
	TS	TSG
Mean	197,3333333	93,22222222
Variance	1061,75	612,9444444
Observations	9	9
Pearson Correlation	0,271986859	
Hypothesized Mean Difference	0	
df	8	
t Stat	8,884510458	
P(T<=t) one-tail	0,0000101875	
t Critical one-tail	1,859548038	
P(T<=t) two-tail	0,0000203749	
t Critical two-tail	2,306004135	

References

[A$^+$13] FIRAS ALGHANIM et al. *Investigating the Impact of Co-located and Distributed Collaboration Using Multi-touch Tables*. PhD thesis, Durham University, 2013.

[AF03] D. Avison and G. Fitzgerald. *Information systems development: methodologies, techniques and tools (3rd edition)*. McGraw-Hill, 2003.

[All13] Wi-Fi Alliance. Wi-fi direct. *URL: http://www. wi-fi. org/discover-wi-fi/wi-fi-direct [accessed: May 2014]*, 2013.

[All15] OSGi Alliance. Open services gateway initiative. *URL: http://www. osgi. org*, 2015.

[AR00] M.B. Albizuri-Romero. A retrospective view of case tools adoption. *ACM SIGSOFT Software Engineering Notes*, 25(2):46–50, 2000.

[BB12] Mohammed Basheri and Liz Burd. Exploring the significance of multi-touch tables in enhancing collaborative software design using uml. In *2012 Frontiers in Education Conference Proceedings*, pages 1–5. IEEE, 2012.

[BD10] B. Bruegge and A.H. Dutoit. *Object-Oriented Software Engineering: Using Uml, Patterns, and Java (3rd Edition)*. Prentice Hall, 2010.

[BFS88] Bonnie Brinton, Martin Fujiki, and Esther A Sonnenberg. Responses to requests for clarification by linguistically normal and language-impaired children in conversation. *Journal of Speech and Hearing Disorders*, 53(4):383–391, 1988.

[BH89] John Brewer and Albert Hunter. *Multimethod research: A synthesis of styles.* Sage Publications, Inc, 1989.

[BM14] Mohammed Basheri and Malcolm Munro. Enhancing the quality of software design through multi-touch interfaces. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, pages 1–7. IEEE, 2014.

[BNT02] Robert Biddle, James Noble, and Ewan Tempero. Lightweight web-based tools for usage-centered and object-oriented design. In *1st International Conference on Usage-Centered Design, Performance-Centered Design, and Task-Oriented Design*, 2002.

[Bol80] Richard A Bolt. *"Put-that-there": Voice and gesture at the graphics interface*, volume 14. ACM, 1980.

[BPL03] Roland Brunken, Jan L Plass, and Detlev Leutner. Direct measurement of cognitive load in multimedia learning. *Educational Psychologist*, 38(1):53–61, 2003.

[CGH03] Qi Chen, John Grundy, and John Hosking. An e-whiteboard application to support early design-stage sketching of uml diagrams. In *Human Centric Computing Languages and Environments, 2003. Proceedings. 2003 IEEE Symposium on*, pages 219–226. IEEE, 2003.

[CL98] N.L. Chervany and D. Lending. Case tools: understanding the reasons for non-use. *ACM SIGCPR Computer Personnel*, 19(2):13–26, 1998.

[CLK$^+$14] Francesco Cafaro, Leilah Lyons, Raymond Kang, Josh Radinsky, Jessica Roberts, and Kristen Vogt. Framed guessability: using embodied allegories to increase user agreement on gesture sets. In *Proceedings of the 8th International Conference on Tangible, Embedded and Embodied Interaction*, pages 197–204. ACM, 2014.

[CNS$^+$95] Joëlle Coutaz, Laurence Nigay, Daniel Salber, Ann Blandford, Jon May, and Richard M Young. Four easy pieces for assessing the usability of multimodal interaction: the care properties. In *Human—Computer Interaction*, pages 115–120. Springer, 1995.

[Con95] Larry L. Constantine. Essential modeling: Use cases for user interfaces. *interactions*, 2(2):34–46, April 1995.

[CR08] Eric Clayberg and Dan Rubel. *eclipse Plug-ins*. Pearson Education, 2008.

[CS91] Paul Chandler and John Sweller. Cognitive load theory and the format of instruction. *Cognition and instruction*, 8(4):293–332, 1991.

[CZ07] Xiang Cao and Shumin Zhai. Modeling human performance of pen stroke gestures. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1495–1504. ACM, 2007.

[DHT00] Christian Heide Damm, Klaus Marius Hansen, and Michael Thomsen. Tool support for cooperative object-oriented design: gesture based modelling on an electronic whiteboard. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 518–525. ACM, 2000.

[DLO09] Bruno Dumas, Denis Lalanne, and Sharon Oviatt. Multimodal interfaces: A survey of principles, models and frameworks. In *Human machine interaction*, pages 3–26. Springer, 2009.

[DVDB03] Gert-Jan De Vreede, Robert M Davison, and Robert O Briggs. How a silver bullet may lose its shine. *Communications of the ACM*, 46(8):96–101, 2003.

[FB99] M. Fowler and K. Beck. *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 1999.

[FCSK03] R. France, S. Chosh, E. Song, and Dae-Kyoo Kim. A metamodeling approach to pattern-based model refactoring. *Software, IEEE*, 20(5):52–58, Sept 2003.

[FHD09] Mathias Frisch, Jens Heydekorn, and Raimund Dachselt. Investigating multi-touch and pen gestures for diagram editing on interactive surfaces. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 149–156. ACM, 2009.

[FR07] Robert France and Bernhard Rumpe. Model-driven development of complex software: A research roadmap. In *2007 Future of Software Engineering*, pages 37–54. IEEE Computer Society, 2007.

[FVdW06] Paul JM Frederiks and Th P Van der Weide. Information modeling: The process and the required competencies of its participants. *Data & Knowledge Engineering*, 58(1):4–20, 2006.

[Gam95] Erich Gamma. *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

[Gre89] Saul Greenberg. The 1988 conference on computer-supported cooperative work: Trip report. *Intelligence*, 1989.

[HC01] George T Heineman and William T Councill. Component-based software engineering. *Putting the pieces together, addison-westley*, page 5, 2001.

[HD06] Tracy Hammond and Randall Davis. Tahuti: A geometrical sketch recognition system for uml class diagrams. In *ACM SIGGRAPH 2006 Courses*, page 25. ACM, 2006.

[Hil00] Rich Hilliard. Ieee-std-1471-2000 recommended practice for architectural description of software-intensive systems. *IEEE, http://standards. ieee. org*, 12:16–20, 2000.

[ISO98] W ISO. 9241-11. ergonomic requirements for office work with visual display terminals (vdts). *The international organization for standardization*, 45, 1998.

[Jac93] Ivar Jacobson. *Object-oriented software engineering: a use case driven approach*. Pearson Education India, 1993.

[JBV$^+$02] Michael Johnston, Srinivas Bangalore, Gunaranjan Vasireddy, Amanda Stent, Patrick Ehlen, Marilyn Walker, Steve Whittaker, and Preetam Maloor. Match: An architecture for multimodal dialogue systems. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 376–383. Association for Computational Linguistics, 2002.

[JCJO92] I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard. *Object-oriented software engineering: a use case driven approach*. Addison-Wesley, 1992.

[Joh88] Robert Johansen. *Groupware: Computer support for business teams*. The Free Press, 1988.

[JOT07] R Burke Johnson, Anthony J Onwuegbuzie, and Lisa A Turner. Toward a definition of mixed methods research. *Journal of mixed methods research*, 1(2):112–133, 2007.

[JS07] A. Jaimes and N. Sebe. Multimodal human-computer interaction: A survey. *Computer Vision and Image Understanding*, 108(1-2):116–134, 2007.

[KH90] Gordon Kurtenbach and Eric A Hulteen. Gestures in human-computer communication. *The art of human-computer interface design*, pages 309–317, 1990.

[KJS11] Eunsuk Kang, Ethan Jackson, and Wolfram Schulte. An approach for effective design space exploration. In *Foundations of Computer Software. Modeling, Development, and Verification of Adaptive Systems*, pages 33–54. Springer, 2011.

[KMB$^+$02] Rilla Khaled, Dan Mackay, Robert Biddle, James Noble, and Ewan Tempero. A lightweight web-based case tool for sequence diagrams. In *Proceedings of the SIGCHI-NZ Symposium on Computer-Human Interaction*, CHINZ '02, pages 55–60, New York, NY, USA, 2002. ACM.

[KRM$^+$13] Dimitrios S Kolovos, Louis M Rose, Nicholas Matragkas, Richard F Paige, Esther Guerra, Jesús Sánchez Cuadrado, Juan De Lara, István Ráth, Dániel Varró, Massimo Tisi, et al. A research roadmap towards achieving scalability in model driven engineering. In *Proceedings of the Workshop on Scalability in Model Driven Engineering*, page 2. ACM, 2013.

[KWBE03] Anneke G Kleppe, Jos Warmer, Wim Bast, and MDA Explained. The model driven architecture: practice and promise, 2003.

[LA97] Jennifer L Leopold and Allen L Ambler. Keyboardless visual programming using voice, handwriting, and gesture. In *Visual Languages, 1997. Proceedings. 1997 IEEE Symposium on*, pages 28–35. IEEE, 1997.

[LL91] Jintae Lee and Kum-Yew Lai. What's in design rationale? *Human–Computer Interaction*, 6(3-4):251–280, 1991.

[LNR$^+$09] Denis Lalanne, Laurence Nigay, Peter Robinson, Jean Vanderdonckt, Jean-François Ladry, et al. Fusion engines for multimodal input: a survey. In *Proceedings of the 2009 international conference on Multimodal interfaces*, pages 153–160. ACM, 2009.

[LP03] S. Lahtinen and J. Peltonen. Enhancing usability of uml case-tools with speech recognition. In *Human Centric Computing Languages and Environments, 2003. Proceedings. 2003 IEEE Symposium on*, pages 227–235. IEEE, 2003.

[LTE$^+$09] Agnes Lanusse, Yann Tanguy, Huascar Espinoza, Chokri Mraidha, Sebastien Gerard, Patrick Tessier, Remi Schnekenburger, Hubert Dubois, and François Terrier. Papyrus uml: an open source toolset for mda. In *Proc. of the Fifth European Conference on Model-Driven Architecture Foundations and Applications (ECMDA-FA 2009)*, pages 1–4. Citeseer, 2009.

[May99] Deborah J. Mayhew. The usability engineering lifecycle. In *CHI '99 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '99, pages 147–148, New York, NY, USA, 1999. ACM.

[MCF03] Stephen J Mellor, Tony Clark, and Takao Futagami. Model-driven development: guest editors' introduction. *IEEE software*, 20(5):14–18, 2003.

[MEH01] Mark W Maier, David Emery, and Rich Hilliard. Software architecture: introducing ieee standard 1471. *Computer*, 34(4):107–109, 2001.

[MFJ93] AC Murray, CR Frankish, and DM Jones. Data-entry by voice: Facilitating correction of misrecognitions. In *Interactive speech technology*, pages 137–144. Taylor & Francis, Inc., 1993.

[MNB03] Daniel Mackay, James Noble, and Robert Biddle. A lightweight web-based case tool for uml class diagrams. In *Proceedings of the Fourth Australasian user interface conference on User interfaces 2003-Volume 18*, pages 95–98. Australian Computer Society, Inc., 2003.

[Moo09] Daniel Moody. The physics of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Trans. Softw. Eng.*, 35(6):756–779, November 2009.

[MVC93] Christophe Mignot, Claude Valot, and Noelle Carbonell. An experimental study of future "natural" multimodal human-computer interaction. In *INTERACT'93 and CHI'93 Conference Companion on Human Factors in Computing Systems*, pages 67–68. ACM, 1993.

[NG47] J. Neumann and HH Goldstine. Planning and coding of problems for an electronic computing instrument. *Institute for Advanced Study, Princeton, New Jersey*, 1947.

[Nie92] Jakob Nielsen. The usability engineering life cycle. *Computer*, 25(3):12–22, 1992.

[Nor86] D.A. Norman. Cognitive engineering. *User centered system design*, pages 31–61, 1986.

[OC00] Sharon Oviatt and Philip Cohen. Perceptual user interfaces: multimodal interfaces that process what comes naturally. *Communications of the ACM*, 43(3):45–53, 2000.

[OCL04] Sharon Oviatt, Rachel Coulston, and Rebecca Lunsford. When do we interact multimodally?: cognitive load and multimodal communication patterns. In *Proceedings of the 6th international conference on Multimodal interfaces*, pages 129–136. ACM, 2004.

[OMGa] OMG. Sysml 1.3 specification. http://www.omg.org/spec/SysML/1.3/. Accessed: 18/09/2012.

[OMGb] OMG. Uml 2.2 specification. http://www.omg.org/technology/documents/formal/uml.htm. Accessed: 18/09/2012.

[OS04] Z. Obrenovic and D. Starcevic. Modeling multimodal human-computer interaction. *Computer*, 37(9):65–72, 2004.

[OV96] Sharon Oviatt and Robert VanGent. Error resolution during multimodal human-computer interaction. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, volume 1, pages 204–207. IEEE, 1996.

[Ovi97] Sharon Oviatt. Multimodal interactive maps: Designing for human performance. *Hum.-Comput. Interact.*, 12(1):93–129, March 1997.

[Ovi99] S. Oviatt. Ten myths of multimodal interaction. *Communications of the ACM*, 42(11):74–81, 1999.

[Ovi03] Sharon Oviatt. Advances in robust multimodal interface design. *IEEE Computer Graphics and Applications*, 23(5):62–68, 2003.

[PCS02] Edwina Pollock, Paul Chandler, and John Sweller. Assimilating complex information. *Learning and instruction*, 12(1):61–86, 2002.

[PF07] Beryl Plimmer and Isaac Freeman. A toolkit approach to sketched diagram recognition. In *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI... but not as we know it-Volume 1*, pages 205–213. British Computer Society, 2007.

[PG90] Richard D. Peacocke and Daryl H. Graf. An introduction to speech and speaker recognition. *Computer*, 23(8):26–33, 1990.

[Pin03] Niels Pinkwart. A plug-in architecture for graph based collaborative modeling systems. In *11th Conference on Artificial Intelligence in Education*, pages 89–94. SIT, 2003.

[PL91] Randy Pausch and James H Leatherby. An empirical study: Adding voice input to a graphical editor. In *Journal of the American Voice Input/Output Society*. Citeseer, 1991.

[RBP$^+$91] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William E. Lorensen, et al. *Object-oriented modeling and design*, volume 199. Prentice-hall Englewood Cliffs, NJ, 1991.

[RC00] K. Renaud and R. Cooper. Feedback in human-computer interaction-characteristics and recommendations. *SOUTH AFRICAN COMPUTER JOURNAL*, pages 105–114, 2000.

[Rit09] Peter Rittgen. Collaborative modeling: A design science approach. In *Hawaii International Conference on System Sciences (HICSS), Hawaii Januari 2009*, pages 1–10, 2009.

[RKdV08] Michiel Renger, Gwendolyn L Kolfschoten, and Gert-Jan de Vreede. Challenges in collaborative modeling: A literature review. In *Advances in Enterprise Engineering I*, pages 61–77. Springer, 2008.

[ROR01] Ronald Rosenfeld, Dan Olsen, and Alex Rudnicky. Universal speech interfaces. *interactions*, 8(6):34–44, 2001.

[RRS$^+$04] Meredith Ringel, Kathy Ryall, Chia Shen, Clifton Forlines, and Frederic Vernier. Release, relocate, reorient, resize: fluid techniques for document sharing on multi-user interactive tables. In *CHI'04 Extended Abstracts on Human Factors in Computing Systems*, pages 1441–1444. ACM, 2004.

[RS05] Nikol Rummel and Hans Spada. Learning to collaborate: An instructional approach to promoting collaborative problem solving in computer-mediated settings. *The Journal of the Learning Sciences*, 14(2):201–241, 2005.

[Rux06] Graeme D Ruxton. The unequal variance t-test is an underused alternative to student's t-test and the mann–whitney u test. *Behavioral Ecology*, 17(4):688–690, 2006.

[SC89] George W Snedecor and Witiiam G Cochran. Statistical methods, 8thedn. *Ames: Iowa State Univ. Press Iowa*, 1989.

[Sch06] Douglas C Schmidt. Model-driven engineering. *COMPUTER-IEEE COMPUTER SOCIETY-*, 39(2):25, 2006.

[Seb09] N. Sebe. Multimodal interfaces: Challenges and perspectives. *Journal of Ambient Intelligence and smart environments*, 1(1):23–30, 2009.

[SHWM05] Heather J Smith, Steve Higgins, Kate Wall, and Jen Miller. Interactive whiteboards: boon or bandwagon? a critical review of the literature. *Journal of Computer Assisted Learning*, 21(2):91–101, 2005.

[SK03] S. Sendall and W. Kozaczynski. Model transformation: the heart and soul of model-driven software development. *Software, IEEE*, 20(5):42 – 45, sept.-oct. 2003.

[SKL$^+$93] Eswaran Subrahmanian, Suresh L Konda, Sean N Levy, Yoram Reich, Arthur W Westerberg, and Ira Monarch. Equations aren't enough: Informal modeling in design. *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*, 7(04):257–274, 1993.

[SMW01] Bernhard Suhm, Brad Myers, and Alex Waibel. Multimodal error correction for speech user interfaces. *ACM transactions on computer-human interaction (TOCHI)*, 8(1):60–98, 2001.

[Sol01] Amy Soller. Supporting social interaction in an intelligent collaborative learning system. *International Journal of Artificial Intelligence in Education (IJAIED)*, 12:40–62, 2001.

[SPH98] Rajeev Sharma, Vladimir I Pavlovic, and Thomas S Huang. Toward multimodal human-computer interface. *Proceedings of the IEEE*, 86(5):853–869, 1998.

[SSL08] Janice Singer, Susan E. Sim, and Timothy C. Lethbridge. *Software Engineering Data Collection for Field Studies*, pages 9–34. Springer London, London, 2008.

[SVMP98] John Sweller, Jeroen JG Van Merrienboer, and Fred GWC Paas. Cognitive architecture and instructional design. *Educational psychology review*, 10(3):251–296, 1998.

[Swe88] John Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive science*, 12(2):257–285, 1988.

[Tur14] Matthew Turk. Multimodal interaction: A review. *Pattern Recognition Letters*, 36:189–195, 2014.

[VDSJM07] Ragnhild Van Der Straeten, Viviane Jonckers, and Tom Mens. A formal approach to model refactoring and model refinement. *Software and Systems Modeling*, 6:139–162, 2007. 10.1007/s10270-006-0025-9.

[VN00] Frederic Vernier and Laurence Nigay. A framework for the combination and characterization of output modalities. In *International Workshop on Design, Specification, and Verification of Interactive Systems*, pages 35–50. Springer, 2000.

[VSBS$^+$94] Maarten W Van Someren, Yvonne F Barnard, Jacobijn AC Sandberg, et al. *The think aloud method: A practical guide to modelling cognitive processes*, volume 2. Academic Press London, 1994.

[Wah03] Wolfgang Wahlster. Towards symmetric multimodality: Fusion and fission of speech, gesture, and facial expression. In *Annual Conference on Artificial Intelligence*, pages 1–18. Springer, 2003.

[WB03] Mike Wu and Ravin Balakrishnan. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 193–202. ACM, 2003.

[WG05] James Wu and TC Nicholas Graham. The software design board: A tool supporting workstyle transitions in collaborative software design. In *Engineering Human Computer Interaction and Interactive Systems*, pages 363–382. Springer, 2005.

[WMW09] Jacob O Wobbrock, Meredith Ringel Morris, and Andrew D Wilson. User-defined gestures for surface computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1083–1092. ACM, 2009.

[WSR$^+$06] Mike Wu, Chia Shen, Kathy Ryall, Clifton Forlines, and Ravin Balakrishnan. Gesture registration, relaxation, and reuse for multi-point direct-touch surfaces. In *First IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP'06)*, pages 8–pp. IEEE, 2006.

[WTF95] Noreen M Webb, Jonathan D Troper, and Randy Fall. Constructive activity and learning in collaborative small groups. *Journal of educational psychology*, 87(3):406, 1995.

[ZLG05] Jing Zhang, Yuehua Lin, and Jeff Gray. Generic and domain-specific model refactoring using a model transformation engine. In *Model-driven Software Development*, pages 199–217. Springer, 2005.