Adoption of Software Modeling Tools: Modeler Experience, Barriers and Benefits

Reyhaneh Kalantari

Thesis submitted to the University of Ottawa
in partial Fulfillment of the requirements for the
Doctorate in Philosophy (PhD) in Digital Transformation and Innovation

School of Engineering Design and Teaching Innovation
Faculty of Engineering

University of Ottawa

Abstract

By providing an abstract view of a complex system, simplifying communication among stakeholders, and generating reliable code, software modeling has been shown to be able to benefit many stakeholders. However, there are ongoing challenges regarding the adoption of modeling.

We present a series of studies about the attitudes that software developers have towards software modeling, and their modeling experience (MX), with some recommendations for improvements.

We followed a multi-method approach, including literature reviews, an interview-based grounded theory study and two surveys. These methods were to some extent iterative, and they informed each other. The earliest literature reviews informed the surveys and the interview study, which in turn informed the later survey and further literature reviews as new themes evolved.

The outcomes of our work can be grouped into three groups of themes. The first group we call mindset barriers. The themes in this group include tooling and support concerns, challenges from modern work practices, behavioural resistance, lack of perceived value and knowledge deficiency. Each of these themes leads us to make various recommendations to modeling tool developers, development managers, and developers themselves. The second group of themes are the perceptions of the benefits of modeling, including improved communication, improved development and improved information representation. These themes can be used to encourage tool developers to focus on the benefits, and to encourage general developers and their management to adopt modeling practices. The third group of themes examines the current practices and utilization of software modeling among developers.

For each of the main themes, we provide both qualitative data, particularly quotations from the interviews and survey comments, and quantitative data. The quantitative data allows us to distinguish how opinions differ depending on the types of modeling tools used, the experience level of developers and other factors.

Acknowledgements

Completing this thesis has been a journey made possible through the support and encouragement of many incredible people.

I am deeply grateful to my supervisor, Professor Timothy Lethbridge, for his exceptional guidance and for standing by me through both the challenges and triumphs of this journey. His patience, insight, and continuous support helped me navigate even the toughest phases, inspiring me to grow as both a researcher and a person.

My appreciation also goes to my committee members, Prof. Jordi Cabot (University of Luxembourg), Prof. Sonia Chiasson (University of Carleton), Prof. Daniel Amyot, and Prof. Shiva Nejati (University of Ottawa) for their time, wisdom, and constructive feedback, which pushed this work to new heights.

To my friends, both here and back home, thank you for being a steady source of laughter, encouragement, and grounding moments that carried me through.

To my family, you have been my foundation and my strength. My dear husband, my best friend forever, Ali Yousefinia, has been my constant rock, offering endless patience and love while providing the emotional support I needed. To my lovely sister, Elham, who inspired me to embark on this PhD journey, your presence here has been my anchor. you have been a continuous source of motivation and joy, inspiring me to move forward each day. And to my parents back home, your support from afar and your belief in my potential have driven me to persevere even in challenging times.

I would like to express my sincere gratitude to NSERC for their financial support, which was instrumental in the completion of this research.

This work is as much yours as it is mine. Thank you all, for being part of this journey.

Reyhaneh Kalantari

Fall 2024

Table of Contents

List of Figures

List of Tables

List of Abbreviations

IMI	Intrinsic Motivation Inventory
MBE	Model-Based Engineering
MBSE	Model-Based Software Engineering
MDD	Model-Driven Development
MDE	Model-Driven Engineering
MX	Modeling Experience
REB	Research Ethics Board
SDFS-2	Short Dispositional Flow Scale
SLR	Systematic Literature Review
TAM	Technology Acceptance Model
UTAUT	Unified Theory of Acceptance and Use of Technology
UX	User Experience

Guide to Colors for Themes Used in Chapters 5, 6, and 7

Topic: Perceived Challenges

Dark Blue	Theme A: Tooling and Support Concerns
Purple	Theme B: Challenges from Modern Work Paradigms
Red	Theme C: Behavioral Resistance
Green	Theme D: Lack of Perceived Value
Yellow	Theme E: Knowledge and Skill Deficiency

Topic: Perceived Benefits

Dark Blue	Theme F: Communication
Green	Theme G: Better Developing
Red	Theme H: Representation of Information

Chapter 1 Introduction

In this thesis we enrich the body of knowledge regarding modeling experience and adoption challenges of software modeling practices. We conduct a variety of empirical studies, accompanied by a theoretical synthesis. We establish a taxonomy representing developers' attitudes towards modeling practices and tools. We believe this will serve as a foundation for improving the modeling experience.

In this chapter, we first provide motivation of our study. Then, we discuss research questions and the research design that we followed, following by the papers that are published from our work and finally present the contributions that we will discuss later in the thesis and the thesis outline.

1.1 Motivation

The core motivations behind our thesis, at its outset, were:

- **Lack of understanding of modeling benefits:** Modeling offers significant benefits, as evidenced by research highlighting clear quality and productivity gains in both industry and education (Dickerson & Mavris, 2013; Holt, 2004; Neto et al., 2019; Gessler, 2019). However, these benefits are clearly not uniformly perceived by the main users, the developers. So, we were motivated to study the perspective of developers and how they perceive and experience these advantages.
- **Lack of adoption of modeling and of an adoption model for modeling:** Despite the benefits, modeling adoption is still limited, and mostly as informal modeling or whiteboard sketches (Akdur et al., 2018; Petre, 2013). We were motivated to deepen the scientific understanding of why this lack of adoption persists within the developer community. By uncovering these reasons, we hoped to identify effective strategies that can promote greater adoption and utilization of modeling techniques.

- **Lack of adoption model for modeling:** The absence of an adoption model for modeling poses a significant challenge for organizations seeking to enhance their use of modeling techniques. While various types of modeling have proven successful across different domains and applications, there is no standardized framework to assess, evaluate, and improve modeling practices within organizations. As highlighted in an insight paper "What Is the Future of Modeling?" by Bucchiarone et al. (Bucchiarone et al., 2021). the need for an adoption model is evident. Such a model could guide organizations through the complexities of implementing and scaling modeling practices. This research is driven by the motivation to address this gap by developing a conceptual framework as a foundation for a well-defined adoption model.
- **Lack of understanding of obstacles to modeling:** Modeling technology has weaknesses that we want to overcome. Research shows that modeling tools and languages have significant obstacles to their use, particularly complexity and lack of analysis/feedback to modelers (Agner et al., 2019; Badreddin et al., 2018; Ozkaya, 2019; Savary-Leblanc, 2019). We were motivated to catalog those obstacles more systematically and explore ways to overcome them. This motivation and the last are tightly correlated, as the lack of use of modeling is influenced by the effectiveness of external factors such as organizational and social issues that we discuss throughout the thesis.
- **Lack of comprehensive research coverage of MX issues:** There are aspects of modeling experience (MX) that have not been systematically studied. As the results of our literature review in Chapter 2 show, there are certain gaps in MX evaluation methods and in the aspects of MX considered so far. We have been motivated to bridge these gaps by taking a more comprehensive view in this thesis.
- **Need for a human-centric approach to modeling tools, as opposed to a technology-centric approach:** The field of software engineering has for many years acknowledged the importance of human factors, recognizing that software development is as much a social activity as a technical one (Lenberg et al., 2015). Human factors such as developers' mindset, their beliefs, attitudes, and previous experiences are crucial elements affecting the adoption of new tools and practices (Wang et al., 2012). There is a growing recognition in the

literature of MBE community (Abrahão et al., 2017), that a shift from a technology-driven approach to a user-driven one is needed in studies. So, this study seeks to undertake an in-depth exploration of the human factors including mindset barriers that hinder software developers from adopting software modeling.

1.2 Research Questions

The main objective of our research is to deepen knowledge of MX. To achieve this, our initial step involves obtaining a comprehensive overview of the current literature on the evaluation of software modeling tools and the reported challenges in their adoption. Subsequently, our focus shifts towards identifying gaps in the existing body of literature, enabling more targeted exploration in those specific domains. Additionally, we aim to capture authentic insights into developers' perceived experiences. In light of these objectives, we frame our main research questions and sub-questions as follows.

1. RQ1. What are the current research practices, evaluation methods, and key trends, findings, and gaps in studying MX and MBE tools?
    1. RQ1-1. What are the trends and themes in publications within this field over time?
    2. RQ1-2- What are the common characteristics of evaluation types in publications?
    3. RQ1-3- Which evaluation methods are commonly employed in publications?
    4. RQ1-4- Which data collection methods are used in publications?
    5. RQ1-5- What is the typical number of participants involved in different types of end-user evaluations?
    6. RQ1-6- What are the professional profiles of participants in user evaluations?
    7. RQ1-7- Which tools and model types were most frequently used in the evaluations?
    8. RQ1-8- How well are modeling tools evaluated for handling large, complex models?
    9. RQ1-9- What are the key challenges of MBE tools identified in publications?

2. RQ2- What are the human-centric barriers to the adoption of software modeling identified in the literature?
3. RQ3- What types of classifications of adoption barriers are reported in the literature?
4. RQ4- How do UmpleOnline (see below) users experience flow while using the platform?
5. RQ5- What are the perceived drivers and barriers to using UmpleOnline among its users?
6. RQ6- How do developers perceive and use software modeling tools in their daily work?
7. RQ7- Is there a relationship between developers' perception of modeling tools and demographic factors such as location, experience, and usage intensity?
8. RQ8- What would be an appropriate conceptual framework that would capture factors contributing to adoption of software modeling tools?

Regarding RQ4, UmpleOnline is a web-based software modeling tool (Lethbridge et al., 2021) developed in the laboratory of the supervisor of this thesis. It is among the few tools that supports editing of models both textually and diagrammatically, while also providing high quality code-generation (Lethbridge et al., 2016). UmpleOnline has a built-in mechanism for prompting users for feedback that any researcher can use. We selected this tool to include in our analysis for its convenience and familiarity, and because the grant funding the research was targeted at improving Umple.

1.3 Research Design

We adopted a comprehensive mixed-methods approach. Our research approach, which is explained in full detail in Chapter 3, employed the following methods:

Literature Analysis: In the initial phase, we conducted a thorough literature analysis. We, of course, built on what others found, particularly studying the considerable differences in their findings. This helped, in part, to achieve all our objectives. We performed several phases of literature analysis, including two systematic reviews, as our research evolved, and new

themes warranted deeper background research. Our literature analyses are the focus of Chapter 2.

Case-Study Analysis: Building upon the insights from the first phase of literature review, we conducted an in-depth examination of the flow experience among developers using the UmpleOnline tool. This analysis served a dual purpose: firstly, it addressed the identified gap in understanding the emotional aspects of the modeling experience, and secondly, it provided critical insights into the usability and effectiveness of UmpleOnline. This is the focus on Chapter 4.

Grounded Theory Exploration: To deepen understanding of developers' attitudes and experiences regarding modeling practices, we utilized grounded theory. The exploratory nature of grounded theory aligns with this research objective. By employing this methodology, we explored and uncovered insights into modeling experience without being constrained by predefined categories or hypotheses, ensuring that the resulting theory is grounded in the lived experiences and perspectives of those involved. We used semi-structured interviews to gain insights of developers' experience with software modeling practices. This is the focus of Chapter 5.

Survey-Based Validation: Subsequently, we transitioned to a survey-based approach to further investigate emerging themes. Engaging a broader spectrum of participants, this method adds a layer of validation to our grounded theory findings, offering a more comprehensive understanding of developers' collective perspectives. These results are reported in Chapter 6.

This mixed-methods strategy, combining literature analysis, case study, grounded theory, and survey-based validation, not only enriches our exploration of modeling practices but also ensures a robust and multifaceted understanding of developers' experiences in this domain.

1.4 Papers Published so Far About this Work

We have published three papers so far:

"Characterizing UX Evaluation in Software Modeling Tools: A Literature Review" (Kalantari & Lethbridge, 2022a): Material from this paper forms the basis of Chapter 2

"Preliminary results of measuring flow experience in a software modeling tool: UmpleOnline." (Kalantari & Lethbridge, 2022b): Material from this paper forms the basis of Chapter 4.

"Unveiling Developers' Mindset Barriers to Software Modeling Adoption," (Kalantari & Lethbridge, 2023): Material from this paper forms the basis of Chapter 5.

1.5 Thesis Contributions

This thesis contributes to the field of software modeling by combining existing knowledge of modeling experience (MX) and modeling adoption with new insights into developers' practices and attitudes towards model-based engineering (MBE) tools. The contributions can be summarized as follows:

1.5.1 Synthesized Landscape of MBE Tool Evaluations

The thesis provides a comprehensive synthesis of the research characteristics of studies involving MBE tool evaluations. This includes trends and themes in publications, evaluation methods, data collection techniques, participant characteristics, prevalent tools, model types, challenges, and less-explored aspects. This collective insight serves as a roadmap for researchers, tool developers, and practitioners, guiding future studies and improvements in MBE tool design.

1.5.2 In-Depth Understanding of Developers' Modeling Perceptions and Practices

Through a detailed investigation, the thesis uncovers the current approaches, attitudes, and preferences of developers towards modeling practices. This includes how developers communicate designs, and tools employed, development stages utilizing modeling, perceived benefits, and challenges. The outcomes contribute to a fine-grained understanding of the dynamics of modeling practices in software development, informing educators, tool developers, and industry professionals.

1.5.3 A Conceptual Framework for Modeling Tool Adoption

Finally, the thesis proposes a conceptual framework that captures the factors contributing to the adoption of software modeling tools. This framework integrates empirical findings and provides a structured approach for analyzing and optimizing factors that influence the adoption and effective use of modeling tools. It lays the groundwork for further research and offers practical implications for enhancing model-based practices in software engineering.

1.6 Thesis Outline

The rest of the thesis is organised according to the following outline:

Chapter 2 Background**:** Discusses the existing research in the domain of software modeling tool evaluations, as well as other information needed to understand the thesis.

Chapter 3 Research Method**:** Discusses the specific procedures used to collect and analyse data during this research.

Chapter 4 Results of Case-Study: An initial survey on flow experience and UmpleOnline tool: Presents the findings and analyzes how users engage with and perceive the tool.

Chapter 5 Results of Grounded Theory**:** Discusses the results of grounded theory conducted in this research, including the coding we developed, backed up by numerous quotations from participants.

Chapter 6 Results of the Survey**:** Presents the detailed findings from the comprehensive survey on developers' modeling practices and perceptions, exploring relationships between different demographic factors and their impact on modeling tool usage.

Chapter 7 Conclusion and Future Work**:** Summarizes the key findings of the research, reflects on the implications of the results, and provides suggestions for future research directions to further explore the adoption and improvement of software modeling tools.

Chapter 2 Background

In this chapter, we review some insights in the form of literature reviews.

Much of the content in this chapter (Section 2.2) is adapted from the systematic literature review that was conducted in 2020 and a paper published in the IEEE Access journal as of December 2022 (Kalantari & Lethbridge, 2022a). As our research evolved, we also found the need to review other related areas of the literature, which are covered in other sections of this chapter.

The literature reviews in this chapter were conducted to answer the sub-research-questions in RQ1.

2.1 Concepts and Definitions

This section explains key terminology used in the background study and summarizes the history of the concepts as discussed in the literature.

2.1.1 Modeling

Models are abstract representations of systems, used to hide details, and decrease the perceived complexity. Booch et al. (Booch et al., 2008) define models as "a simplification of reality that is created in order to better understand the system, as we cannot comprehend complex systems." Models organize and represent information about the system and its interactions with the environment (Madni & Sievers, 2018).

Selic (Selic, 2016) defines an engineering model as "a selective representation of some system intended to capture accurately and concisely all of its essential properties of interest for a given set of concerns."

Models have been used in software development from the early years of computer systems (Dickerson & Mavris, 2013), and many practitioners know them as a key factor in achieving software project success, since they empower both engineering and communication aspects of the software process.

A model can be depicted through various means, including using modeling languages such as state machines, algorithms, equations, and parametric curves (Madni & Sievers, 2018). This approach offers a unified view of modeling and programming and advocates for the perspective that "programs are models", where traditional code is considered as just a less-abstract model (Haxthausen & Peleska, 2016). By treating software programs as models, this perspective bridges the gap between software development and systems modeling, facilitating a more holistic view of both disciplines.

2.1.2 Terminology and Acronyms in Modeling

In the literature, various terminology is used in the context of software modeling, such as MDE, MBE, MDD, and others. These terms are sometimes used interchangeably. In our work, we adopt the terminology proposed by Brambilla et al.(Brambilla et al., 2017). Figure 1 shows the hierarchical relationship between common terms in modeling.

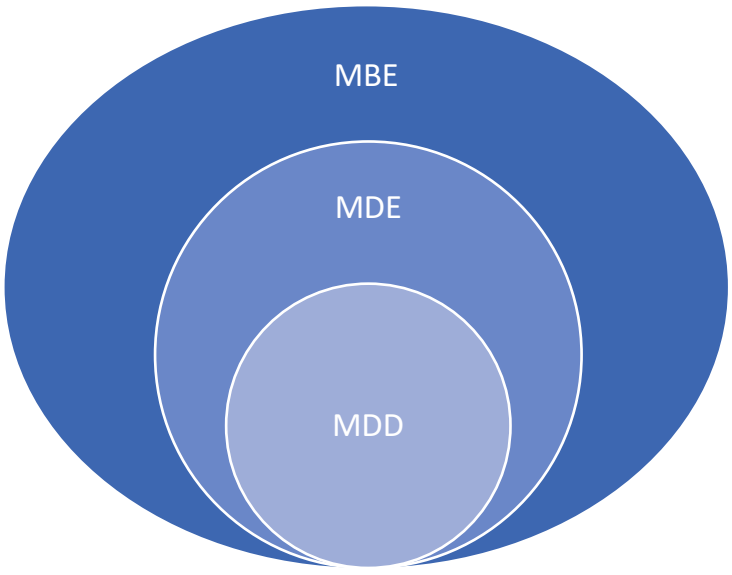


Figure 1. Relationship between MDD,MDE, and MBE (Brambilla et al., 2017)

Model-Driven Development (MDD) is a development paradigm where models are the primary artifacts, and implementation is often (semi)automatically generated from these models.

Model-Driven Engineering (MDE) is broader than MDD, encompassing additional model-based tasks beyond development, such as system evolution, analysis and reverse engineering. In other words, whereas 'Development' means just building software, 'Engineering' implies also focusing on economics, quality, change management and similar issues.

Model-Based Engineering (MBE), or model-based development, refers to processes where models play an important but not central role. Unlike MDE, models in MBE are not necessarily the key artifacts driving the process; for example, they may serve as sketches 'blueprints' for manual coding without automatic code generation; or they may be used for only certain parts of the system.

Since we are focusing on software, it's important to note that terms like 'development' and 'software development' can be used interchangeably, implying building software systems. The above three acronyms sometimes have an S injected after the second letter, so MBE can also be, in our domain, called MBSE. We will omit the S in this thesis and consider MBE and MBSE to be the same for our purposes.

Given these distinctions, this thesis will focus on Model-Based Engineering (MBE) in the context of software, acknowledging that 'software' is implied unless specified otherwise. This approach is broad and inclusive, allowing us to consider various methodologies and formality levels under the MBE umbrella.

2.1.3 Different Software Modeling Categorization

There are different types of categorizations in the literature for modeling tools, modeling languages, or modeling in general.

A common categorization is based on the formality of modeling languages: *informal* models include text descriptions, concept diagrams, or stylized drawings without formal semantics, sometimes in the style of cartoons. In contrast, *formal* models utilize structured formalisms, such as state charts, sequence diagrams, or Petri nets, along with their visual representations (Madni & Sievers, 2018). The most formal models can be mathematically or logically analysed by software; the term 'formal methods' refers to these types of models.

However, we also consider models that can be analyzed by a computer to provide insights into a design, generate code, or produce other models, even if they exhibit a lower level of formality.

Additionally, models can be categorized by their users' goals: facilitating communication, capturing and documenting designs, enabling analysis and understanding, generating code, and conducting simulations (Savary-Leblanc et al., 2024).

Another categorization differentiates between modeling tools that support *graphical* representations, those that support both graphical and *textual* representations (Madni & Sievers, 2018), and those that support just textual representations. In some literature there is an implicit assumption that models are graphical, but we do not take this perspective.

Another common categorization is based on the intended purpose of the model: *descriptive* models are used to represent the reality of a system or context and outline the scope and specifics for studying a problem, and *prescriptive* models define how a system should be implemented (Brambilla et al., 2017) "Typically prescriptive models are defined at a low-level of abstraction since they should contain the information that is needed to implement the system (Heldal et al., 2016)". These models serve not only in code generation but also in generating configuration files, test cases, and various other non-code artifacts vital for product development (Heldal et al., 2016). We note that some kinds of prescriptive models can actually be at a reasonably high level of abstraction (e.g. certain state machines that can be used to generate code), but they would need to be combined with other information, independent of the model, to describe how the abstract model should be implemented.

Brown (Brown, 2004) suggest that the current modeling practices range from a code-only approach, where developers rely solely on code within Integrated Development Environments (IDEs), to using graphical notations for code visualization, aiding understanding and manipulation of code structures. The visualization is shown in Figure 2.

Another practice that has been tried is roundtrip engineering (RTE) (Hettel et al., 2008), where abstract system models are iteratively transformed into detailed design models and code, and any changes to the code are reflected back into the model so that further changes to the model are then reflected in new code; this proves complex and requires sophisticated synchronization between models and implementations.

Model-centric approaches allow complete system implementations to be generated from detailed models, often using application frameworks and runtime services.

At the far end of the spectrum, a model-only approach uses models for understanding, discussion, and analysis without direct linkage to system implementation.

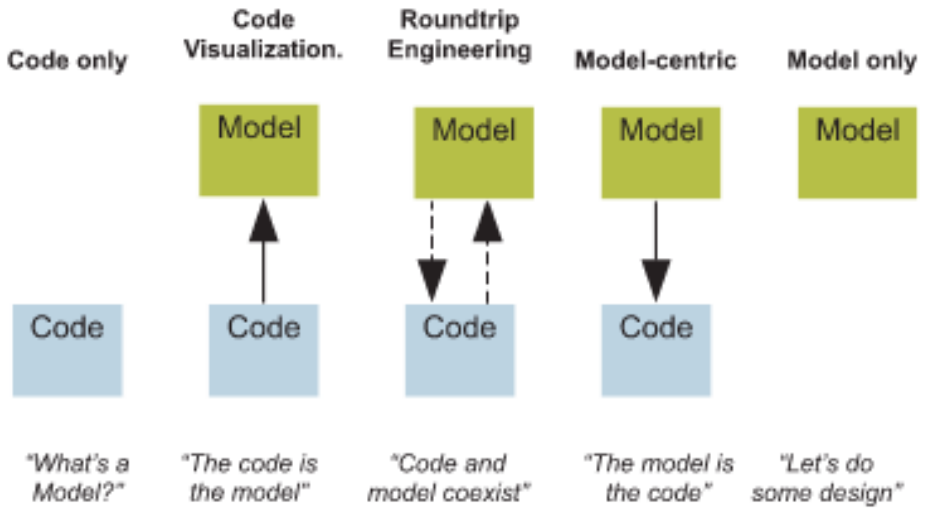


Figure 2: Modeling Spectrum (Brown, 2004)

Bucchiarone et al. (Bucchiarone et al., 2021), in their insight paper about the future of modeling, talk about three successful forms of modeling: model-based systems engineering (MBSE), low-code software development, and informal software modeling.

2.1.4 User Experience

The term "User Experience" (UX) was first introduced by Norman (D. Norman, 1988), and the interest of practitioners and researchers in the concept increased when they became convinced that a usability framework focusing only on user performance has limitations (Law et al., 2009), since there is a need to satisfy users' growing expectations. UX covers various concepts, from traditional usability (focusing on efficiency of use, learnability, and error prevention) to emotional, experiential, hedonic, and aesthetic variables (Hassenzahl & Tractinsky, 2006).

Hassenzahl and Tractinsky (Hassenzahl & Tractinsky, 2006) define UX as, "consequence of a user's internal state (predispositions, expectations, needs, motivation, mood, etc.), the characteristics of the designed system (e.g., complexity, purpose, usability, functionality, etc.) and the context (or the environment) within which the interaction occurs (e.g., organizational/social setting, the meaningfulness of the activity, voluntariness of use, etc.)."

Zarour and Alharbi (Zarour & Alharbi, 2017), proposed a framework consisting of UX dimensions, aspects categories, and measurement methods. They categorized UX into four dimensions including Value, NX (need experience), BX (brand experience), and TX (technological experience), and then provided corresponding aspects and methods for each dimension. In another study, Zarour (Zarour, 2020), developed a user experience needs evaluation method mapping the quality factors introduced in the previous paper with ISO 25000 standard factors for software products, and using evaluation theory. The criteria in his evaluation method are "usefulness", "pleasure", "aesthetics", and "trust".

Law et al. (Law et al., 2009) investigated UX's scope by surveying 275 domain researchers and practitioners and concluded that UX is dynamic, context-based, and inherently subjective, which could be defined by interacting with a product, system, service, or object. As a result of these characteristics, the topic has become controversial, with much effort being devoted to describing and defining its scope in different contexts.

We intend to define and characterize UX in one specific context: the domain of *modeling*. We refer to Hartson and Pyla's definition (Hartson & Pyla, 2019):

"User experience is the totality of the effects felt by the user before, during, and after interaction with a product or system in an ecology."

We base our study on the UX components presented in Hartson and Pyla's work (Hartson & Pyla, 2019)*:* usability, usefulness (utility), emotional impact, and meaningfulness.

2.1.5 User Experience in MBE tools

Some studies point to bad user experience or certain factors of user experience as key barriers to adoption of modeling tools (Agner & Lethbridge, 2017; Planas & Cabot, 2020; Vogelsang et al., 2017; Weber, Zoitl, & Hubmann, 2019). However, we could not find much

research focusing on all the relevant aspects of user experience in modeling tools or provide any comprehensive guideline to MX evaluation. As discussed in the previous section, user experience is a broad concept covering a range of issues, which needs a definition based on the context. Abrahão et al. (Abrahão et al., 2017) are among pioneering researchers in this domain who introduced the term 'MX' and emphasized the need for more theoretical and empirical research to define the user experience in modeling tools. We aim to build on their work, gathering and analyzing existing literature that addresses the quality, usability, and user experience evaluation of modeling tools.

2.1.6 Theoretical Frameworks: The Technology Acceptance Model (TAM) and its UTAUT Enhancement

One of the objectives of our research is understanding the factors influencing the adoption of software modeling tools. The Technology Acceptance Model (TAM), introduced by Davis (Davis, 1986), provides a foundational framework for this. TAM suggests that Perceived Usefulness and Perceived Ease of Use are the primary determinants of an individual's intention to use a technology, which subsequently predicts actual usage.

Building on TAM, the Unified Theory of Acceptance and Use of Technology (UTAUT), developed by Venkatesh et al. (Venkatesh et al., 2003), offers a more comprehensive approach by incorporating elements from eight different models of technology acceptance. UTAUT identifies four key constructs – *performance expectancy*, *effort expectancy*, *social influence*, and *facilitating conditions* – that influence *behavioral intention* and *usage behavior*. Additionally, UTAUT considers the impact of moderating variables such as gender, age, experience, and voluntariness of use. Figure 3 presents this model.

We found this model as particularly useful for presenting the broader organizational and social factors that affect the adoption of software modeling tools among developers, as explored in this thesis.

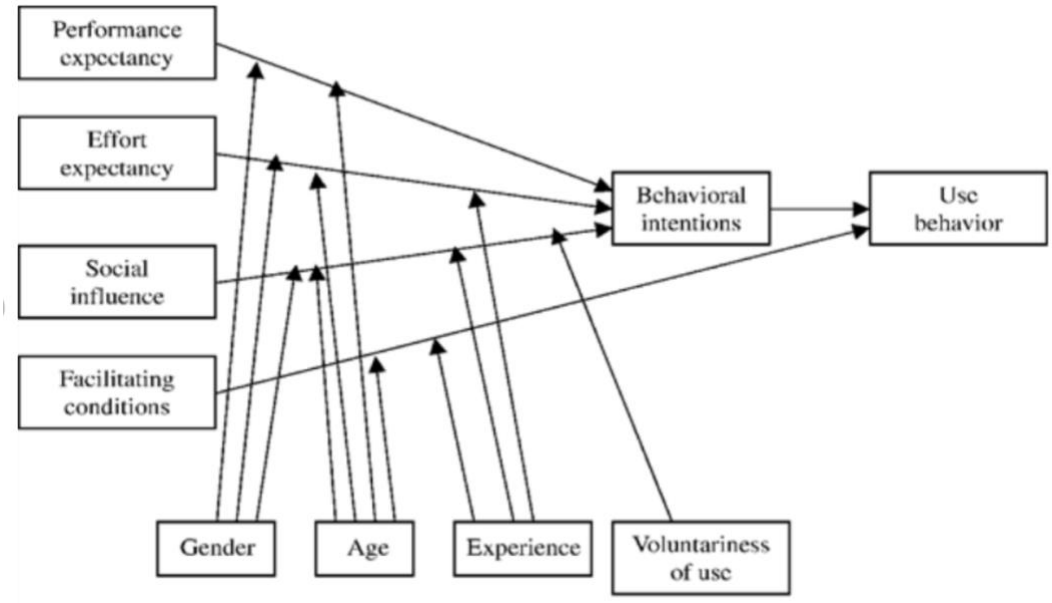


Figure 3. UTAUT Model (Venkatesh et al., 2003)

2.2 Systematic Literature Review about Modeling Tools

In 2020, we performed a systematic literature review to comprehensively analyze and evaluate existing research in the field. This is the first of two SLRs described in this chapter. The following subsections will outline the key methodologies employed, highlight the main findings, discuss notable trends identified, and conclude with implications for future research and potential areas for further investigation.

2.2.1 Methodology of the SLR

This study aims to gain a better understanding about MBSE tool evaluation as well as the known problems and challenges in MX. Therefore, we applied a systematic literature review (SLR) method, whose objective is to evaluate and summarize published information to address issues regarding the subject in an unbiased way. Kitchenham (B. Kitchenham, 2004) defines a systematic review as a "means of identifying, evaluating, and interpreting all available research relevant to a particular research question, topic area, or phenomenon of interest. Individual

studies contributing to a systematic review are called primary studies; a systematic review is a form a secondary study."

Undertaking an SLR based on searching in databases is common in many disciplines. However, it has some challenges, including the need to formulate good expressions (the key activity of database searches), the different interfaces to the databases, and differing limitations of the databases (Wohlin, 2014).These challenges become highlighted when searching in a domain that does not have a standard terminology to use as keywords in the search query, or where the vocabulary uses very general words.

We benefited from the "guidelines for systematic literature reviews" provided by Wohlin (Wohlin, 2014), which describes *snowballing* as a good research approach for SLR in software engineering and formulates the steps of its procedure. Figure 4 demonstrates the procedure of snowballing.

When applying snowballing, one first identifies a *start set* of papers (Wohlin, 2014). With the start set, one then uses backward and forward snowballing. "Backward snowballing means using the reference list to identify new papers to include, and forward snowballing refers to identifying new papers based on those papers citing the paper being examined. After backward and forward snowballing, new papers identified in the iteration are put into a pile to go into the next iteration" (Wohlin, 2014). The iterations continue until no more new papers are found.

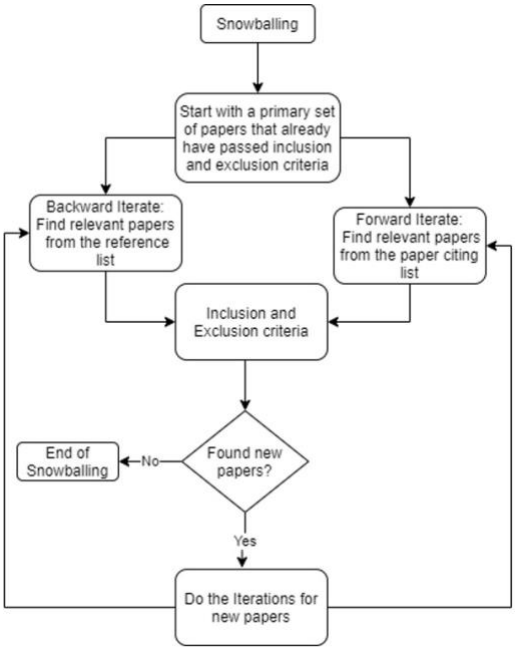


Figure 4: Snowballing procedure for our SLR about UX of modeling tools

2.2.1.1 Search Process

We used Google Scholar to identify the start set, which Wohlin's (Wohlin, 2014) recommends as a good choice to avoid publisher bias. Since we are using snowballing, we did not need to craft a comprehensive search query; we only needed to find a few papers that would be among the interconnected set of relevant papers.

We initiated the search process by using the expression below, derived from keywords in our research questions.

"(user experience OR UX) evaluation in (software modeling tools OR MBSE OR MDE)"

This query retrieved around 2 million results in Google Scholar and presented them ordered by the search algorithm's computation of relevance. To further assure relevance, we then manually screened the first 30 papers. Had we screened more papers, we would likely have ended up with fewer snowballing iterations; had we screened fewer, we would likely have needed more snowballing iterations.

For each paper, we first screened based on its title, applying the inclusion and exclusion criteria presented in the next section. For any paper that was not yet excluded, we went further and read its abstract. If we still could not decide about its relevance, we read the whole paper to decide whether to include it.

Following the above screening, we obtained a primary set of six most-relevant papers, which we have marked with an asterisk (*) in Table 2. Backward and forward snowballing in the first iteration gave 364 results, of which 18 papers were selected to include in our pool. In the second iteration, we found 1183 papers, among which 13 papers were included. The next iteration was performed with 606 papers resulting from snowballing, and we selected 4 more relevant papers considering our inclusion and exclusion criteria. For the last iteration, from 169 papers, we could not find any new relevant papers for our study. Therefore, the snowballing procedure for this study was finished by finding 41 papers for further analysis. The process is shown in Figure 5.

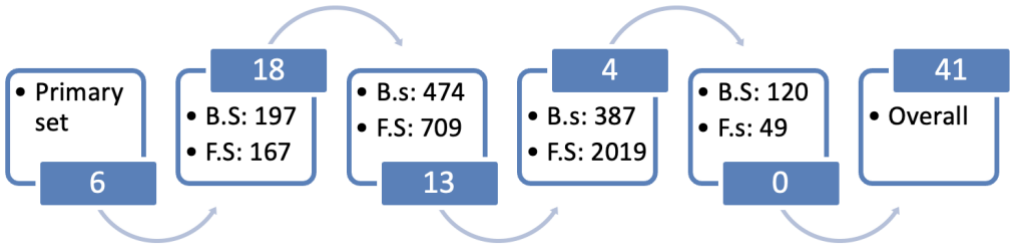


Figure 5: Paper counts in the SLR after each backward snowballing (B.S.) and forward snowballing (F.S.) step

2.2.1.2 Inclusion and Exclusion Criteria

We selected references based on the inclusion and exclusion criteria stated in Table 1. The criteria are constructed based on the research goals and questions.

Table 1: Inclusion and exclusion criteria for our SLR about UX of modeling tools

Inclusion Criteria	Exclusion Criteria
Papers in the English language	Papers that only present specific feature requirements for creating MBE tools
Papers published between the years 2000-2020 inclusive.	Papers that present adoption issues beyond MBE tool usage like modeling language issues
Papers that highlight MBE tool usage issues by evaluation	Grey literature (theses, non-refereed papers, manuals, books)
Papers that evaluate MBE tools	

2.2.1.3 Data Extraction

We developed a template based on our research questions and entered all relevant information about each resulting paper. The extraction process included the following information:

- Title
- Publication type
- Publication year
- Goal of study
- Focus of evaluation
- Evaluation method
- Data collection tools
- The number and types of assessed tools and diagrams
- The average number of participants in user testing
- Participants' profiles
- Size of evaluated models
- Challenges or issues

2.2.1.4 SLR Data Analysis Process

We employed a combination of quantitative and qualitative methods for data analysis. Descriptive statistics were utilized to outline the characteristics of studies in our domain,

encompassing evaluation type, method, participant profile, tool type, and more. Simultaneously, content analysis was applied to categorize reported modeling experience (MX) issues and construct a taxonomy of tool-related issues. Content analysis was chosen for its flexibility and systematic approach in identifying themes and patterns in qualitative datasets (Saunders et al., 2019).

In our coding process, we implemented directed coding to align our findings with a pre-existing user experience model (Hartson & Pyla, 2019) explained above in Section 2.1.4, subsequently customizing it to encompass all factors identified in our literature review. Directed coding is particularly useful when prior research exists about a phenomenon that requires further description or extension (Hsieh & Shannon, 2005).

We used Excel spreadsheets in the organizing and coding our data. Part of the larger MS-Excel screenshot of our categorizing process of papers is shown in Figure 6.

Figure 6: Part of the larger MS-Excel screenshot of organizing and categorizing literature in the first SLR

2.2.2 SLR Results

This section summarizes and categorizes the findings.

The list of selected papers based on their publication year, publication type, approach, and goal are shown in Figure 6. The papers are organized in this and subsequent tables using

reverse chronological order. *Journal* articles are tagged with 'J' in the third column of the table, and *Conference* articles, including workshops and symposia, are tagged 'C'.

We identified several primary approaches taken by the papers, as follows: The code in brackets at the start of each item, is used to mark the tool in the 'Approach' Column of Table 2.

- (EF) To propose an *evaluation framework* or template, usually alongside a case study to validate that framework.
- (NF) Proposing a *new feature* for improving MBSE tools and evaluating tools to validate the suitability of the feature.
- (CE) To *comparatively evaluate* several tools.
- (W) Presenting the current state and trend of MBSE tools, regarding their *weaknesses* and adoption issues.

Table 2: List of selected papers for the first SLR with the primary set flagged with asterisks

Ref.	Conf./ Journal	Title	Approach	Goal of study
(Ozkaya & Erata, 2020)	C	Understanding Practitioners' Challenges on Software Modeling: A Survey	W	Understand the challenges that practitioners face
(Planas & Cabot, 2020)	C	How are UML class diagrams built-in practice? A usability study of two UML tools: Magicdraw and Papyrus	CE	Draw some useful lessons that help to improve the UML modeling process
(Ren et al., 2020)	C	Collaborative Modelling: Chatbots or On-Line Tools? An Experimental Study	NF	Evaluate the suitability of chatbots for collaborative modeling
(Bucchiarone, Savary-Leblanc, et al., 2020)	C	Papyrus for gamers, let's play modeling	NF	Use gamification to help students learning to model
(Ozkaya, 2019)	C	Are the UML modeling tools powerful enough for practitioners? A literature review	CE	Analyze a set of existing UML modeling tools
*(Agner et al., 2019)	C	Student experience with software modeling tools	CE	Help professors and students choose tools based on their strengths and weaknesses
*(Weber, Zoitl, & HuBmann, 2019)	C	Usability of Development Tools: A CASE-Study	EF	Test and analyze several evaluations methods

Ref.	Conf./ Journal	Title	Approach	Goal of study
(Stegmaier et al., 2019)	C	Insights for Improving Diagram Editing Gained from an Empirical Study	W	Identify how diagram editing can be improved by investigating how people model on whiteboards
*(Savary-Leblanc, 2019)	C	Improving MBSE Tools UX with AI-Empowered Software Assistants	NF	Implement software assistants in a modeling tool
(Badreddin et al., 2018)	C	A Decade of Software Design and Modeling: A Survey to Uncover Trends of the Practice	W	Uncover trends and the adoption patterns of modeling languages such as UML
(Robles Luna et al., 2018)	C	Challenges for the adoption of model-driven web engineering approaches in industry	W	Present the current problems of MDE approaches
(Mert Ozkaya & Ferhat Erata, 2018)	C	Analyzing UML Modeling Tools for Practical Use	CE	Assess tools' support for requirements
(Pietron et al., 2018)	C	A Study Design Template for Identifying Usability Issues in Graphical Modeling Tools	EF	Present a study design template for identifying usability issues in graphical editors
*(Pourali, 2018)	C	An Empirical Investigation to Understand the Difficulties and Challenges of Sofware Modellers When Using Modelling Tools	W	Identify the most prominent difficulties that users might face when using UML modeling tools
(Störrle, 2018)	C	Improving model usability and utility by layered diagrams	NF	Understand how layered diagrams are useful in modeling tools
(Agt-Rickauer et al., 2018)	C	DoMoRe – A Recommender System for Domain Modeling	NF	Support the domain modeling process by implementing an automated modeling recommendation
(Akdur et al., 2018)	J	A survey on modeling and model-driven engineering practices in the embedded software industry	W	Investigate practices in the embedded software engineering projects
(Liebel et al., 2018)	J	Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice	W	Assess the current state-of-practice and the challenges of embedded systems domain are facing due to shortcomings with MBE
(Agner & Lethbridge, 2017)	C	A Survey of Tool Use in Modeling Education	CE	Evaluate tool support for teaching modeling
(Whittle et al., 2017)	J	A taxonomy of tool-related issues affecting the	W	Present a taxonomy of tool-related considerations

Ref.	Conf./ Journal	Title	Approach	Goal of study
		adoption of model-driven engineering		
(Vogelsang et al., 2017)	C	Should I Stay or Should I Go? On Forces that Drive and Prevent MBSE Adoption in the Embedded Systems Industry	W	Investigate issues of MBSE adoption in embedded software developing companies
(Bordeleau et al., 2017)	C	Challenges and Research Directions for Successfully Applying MBE Tools in Practice	W	Discuss challenges in applying MBE from academic and industrial viewpoints
(Ribeiro et al., 2016)	C	Comparative analysis of workbenches to support DSMLs: Discussion with non-trivial Model-Driven Development needs	CE	Analysis and comparison of three tools
*(Safdar et al., 2015)	C	Empirical Evaluation of UML Modeling Tools–A Controlled Experiment	CE	Compare the productivity of the software engineers while modeling with the tools
(Ahmar et al., 2015)	C	Enhancing the communication value of UML models with graphical layers	NF	Using the layers in UML diagram editors
(Ferreira et al., 2014)	C	Characterizing the Tool-notation-people Triplet in Software Modeling Tasks	EF	Present a usability evaluation template and evaluate two tools based on that
(Williams et al., 2014)	C	Software Analytics for MDE Communities	EF	Propose to apply software analytics techniques on open source MDE tools and languages
(Whittle et al., 2013)	C	Industrial Adoption of Model-Based Engineering: Are the Tools Really the Problem?	W	Present a taxonomy of tool-related issues
(Condori-Fernández et al., 2013)	J	An empirical approach for evaluating the usability of model-driven tools	EF	Present a usability evaluation template
(Kuhn et al., 2012)	C	An Exploratory Study of Forces and Frictions Affecting Large-Scale Model-Driven Development	W	Investigate model-driven engineering
(El Kouhen et al., 2012)	C	Evaluation of Modeling Tools Adaptation	CE	Review tool's adaptation approaches
(Heena & Ranjna, 2011)	C	A comparative study of UML tools	CE	Compare most-used tools based on some features
(Panach et al., 2011)	C	An Experimental Usability Evaluation Framework for Model-Driven Tools	EF	Present a usability evaluation template

Ref.	Conf./ Journal	Title	Approach	Goal of study
(K.-H. Huang et al., 2011)	C	Hammering Models: Designing Usable Modeling Tools	EF	Facilitate improvement of modeling tools by identifying common problems in existing tools and developing a framework of redesign guidelines
(Eichelberger et al., 2009)	C	A Comprehensive Survey of UML Compliance in Current Modelling Tools	CE	Asses the UML compliance levels of modeling tools
(Saraiva & Silva, 2008)	C	Evaluation of MDE Tools from a Metamodeling Perspective	EF	Present an evaluation framework for tool support of the metamodeling activity, and evaluate a small set of tools
*(Auer et al., 2007)	C	Explorative UML modeling comparing the usability of UML tools	CE	Assess tools' usability for UML sketching
(Störrle, 2007)	C	Large Scale Modeling Efforts: A Survey on Challenges and Best Practices	W	Present challenges in large modeling
(Egyed, 2006)	C	Instant consistency checking for the UML	NF	Introduce an approach for quickly, correctly, and automatically deciding when to evaluate consistency rules in modeling
(Lahtinen & Peltonen, 2005)	C	Adding speech recognition support to UML tools	NF	Present an approach to develop speech interfaces in UML tools
(Seffah & Rilling, 2001)	C	Investigating the relationship between usability and conceptual gaps for human-centric CASE tools	W	Highlight common usability problems in the most popular Java IDEs

2.2.2.1 RQ1-1- What are the trends and themes in publications within this field over time?

The bar chart in Figure 3 illustrates the number of published papers over a period of 20 years, in different categories.

An overview of selected papers shows that the number of papers rises after 2016 and reaches the highest point in 2018, with nine papers. The decreasing trend after 2018 is probably due to the delay in publishing papers in the most recent years, and the fact that backward snowballing to recent papers is less likely than it is to earlier papers. It is expected to continue

rising in the future, especially because usability and user experience concepts are emerging fields that attract more and more attention from academia and industry.

As Figure 7 shows, most papers, including the oldest one, belong to the category W (Weaknesses). This highlights that investigating weaknesses and adoption issues of MBSE tools, is among the most recurring concerns of researchers in this domain.

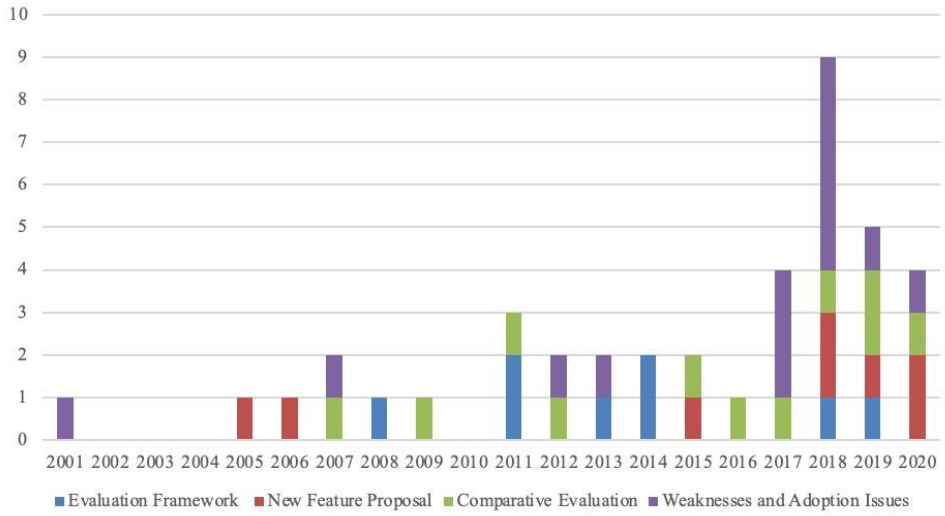


Figure 7: Paper publishing trend in the papers selected in the first SLR

2.2.2.2 RQ1-2- What are the common characteristics of evaluation types in publications?

We describe the characteristics of each evaluation using the set of criteria that form the column headings of Table 3. These criteria are the a) focus of the evaluation, b) the data collection methods used (inquiry, inspection and testing), c) the background of tool-user participants when such participants were involved (students, professors, modelers in industry, or a combination of these groups), d) the number of MBSE tools which are assessed by the paper, e) whether the paper lists challenges, f) whether it suggests improvements, and g) the specific suggested improvements (if any).

The focus of evaluation in some cases was usability criteria such as efficiency; in other cases, it was assessing certain requirements, or participants' behavior and experience. Two

studies assessed more than 50 tools using the inspection method performed by researchers (Ozkaya, 2019),. As illustrated in Table 3, 85% of papers list some challenges and issues, while 51% suggest improvements, either theoretically or by implementing a particular feature including gamification (Bucchiarone, Savary-Leblanc, et al., 2020), AI-empowered software assistants (Savary-Leblanc, 2019), layered diagrams (Ribeiro et al., 2016; Störrle, 2018), instant consistency checking (Egyed, 2006), and speech recognition (Lahtinen & Peltonen, 2005).

Table 3: Selected papers in the first SLR, and their properties

Ref.	Focus of evaluation	Data collection methods	End-user participant types	Number of tools assessed	List challenges	Improvement suggestions	Suggested improvement feature
(Ozkaya & Erata, 2020)	Eight categories of software modeling challenges	Inquiry	Industry		Yes		
(Planas & Cabot, 2020)	Efficiency	Inspection, Testing	Academia	2	Yes	Yes	
(Ren et al., 2020)	Efficiency, effectiveness, satisfaction	Testing, Inquiry	Academia	2			
(Bucchiarone, Savary-Leblanc, et al., 2020)	-			1	Yes	Yes	Gamification
(Ozkaya, 2019)	Specific requirements	Inspection		58	Yes	Yes	
(Agner et al., 2019)	Learnability	Inquiry	Academia	31	Yes	Yes	
(Weber, Zoitl, & HuBmann, 2019)	Usability criteria	Testing, Inquiry, Inspection	Industry	1			
(Stegmaier et al., 2019)	Behavior of users	Testing, Inquiry	Academia		Yes	Yes	
(Savary-Leblanc, 2019)	-				Yes	Yes	AI-Empowered Software Assistants
(Badreddin et al., 2018)	Practitioners' experience	Inquiry	Multi experience level		Yes		
(Robles Luna et al., 2018)	Specific requirements	Inquiry	Industry		Yes		

Ref.	Focus of evaluation	Data collection methods	End-user participant types	Number of tools assessed	List challenges	Improvement suggestions	Suggested improvement feature
(Mert Ozkaya & Ferhat Erata, 2018)	Specific requirements	Inspection		11	Yes		
(Pietron et al., 2018)	Usability of graphical editors	Testing, Inquiry	Experts	1			
(Pourali, 2018)	Efficiency, effectiveness, satisfaction	Testing, Inquiry	Academia Industry		Yes	Yes	
(Störrle, 2018)	Validation of a new feature	Testing, Inquiry	Academia Industry	1	Yes	Yes	Layered diagrams
(Agt-Rickauer et al., 2018)	Validation of a new feature		Industry		Yes	Yes	Automated modeling recommendation
(Akdur et al., 2018)	Latest trends in modeling	Inquiry	Academia Industry	10	Yes	Yes	
(Liebel et al., 2018)	Practitioners' experience	Inquiry	Industry	3	Yes		
(Agner & Lethbridge, 2017)	Professors' experience in teaching using modeling tools	Inquiry	Academia	32	Yes	Yes	
(Whittle et al., 2017)	Practitioners' experience	Inquiry	Industry		Yes	Yes	
(Vogelsang et al., 2017)	Practitioners' experience	Inquiry	Industry		Yes		
(Bordeleau et al., 2017)	Experience in industry	Inspection			Yes		
(Ribeiro et al., 2016)	Suitability for developing non-trivial language	Inspection		3	Yes		
(Safdar et al., 2015)	Productivity (completeness, memory load, learnability, and modeling effort)	Testing, Inquiry	Academia	3	Yes		
(Ahmar et al., 2015)	Validation of new feature(layers)	Inspection		1	Yes	Yes	Add layers
(Ferreira et al., 2014)	HCI perspectives: Usability,Communicability	Inspection		1	Yes		

Ref.	Focus of evaluation	Data collection methods	End-user participant types	Number of tools assessed	List challenges	Improvement suggestions	Suggested improvement feature
(Williams et al., 2014)	MDE community	Inspection		10			
(Whittle et al., 2013)	Practitioners' experience	Inquiry	Industry		Yes	Yes	
(Condori-Fernández et al., 2013)	Efficiency, effectiveness, satisfaction	Testing	Multi experience level	1	Yes		
(Kuhn et al., 2012)	Practitioners' experience	Inquiry	Industry		Yes		
(El Kouhen et al., 2012)	Customizability and usability	Inspection		5			
(Heena & Ranjna, 2011)	Specific requirements	Inspection		6	Yes	Yes	
(Panach et al., 2011)	Effectiveness and efficiency	Testing	Multi experience level	1	Yes	Yes	
(K.-H. Huang et al., 2011)	Behavior and experience of users	Inspection, Testing, Inquiry	Industry	1	Yes	Yes	
(Eichelberger et al., 2009)	Specific requirements	Inspection		68	Yes		
(Saraiva & Silva, 2008)	Metamodeling activity	Inspection		4	Yes	Yes	
(Auer et al., 2007)	Efficiency, effectiveness	Inspection		2	Yes	Yes	
(Störrle, 2007)	Practitioners' experience	Inquiry	Industry		Yes		
(Egyed, 2006)	Validation of a new feature	Inspection		1	Yes	Yes	Instant consistency checking
(Lahtinen & Peltonen, 2005)	Behavior and experience of users	Testing, Inquiry	Novices	1	Yes	Yes	Speech recognition
(Seffah & Rilling, 2001)	Ease of use	Inquiry	Industry	5	Yes	Yes	

2.2.2.3 RQ1-3- Which evaluation methods are commonly employed in publications?

Evaluation methods in the reviewed publications covered a wide range of methodologies with different combinations, some of which are more concerned with certain features and functionality of tools, while others emphasize user-centered evaluations.

From 41 papers, 38 papers mentioned their methods explicitly. We categorized these papers based on their employed evaluation method. As shown in Figure 8, more than half of the publications (58%) applied user-centered evaluation, with the user types shown in the 'End-user participant types' column of Table 3. For many other papers (34%) it was the researchers themselves that performed the evaluations, mostly based on feature checklists (with the value 'Inspection' of the 'Data Collection Methods' column of Table 3). Only two papers (5%) leveraged both researcher and tool-user evaluation methods, and just one paper (Weber, Zoitl, & HuBmann, 2019), benefitted from UX expert feedback besides user evaluation, and it was concerned with presenting different usability evaluation methods.

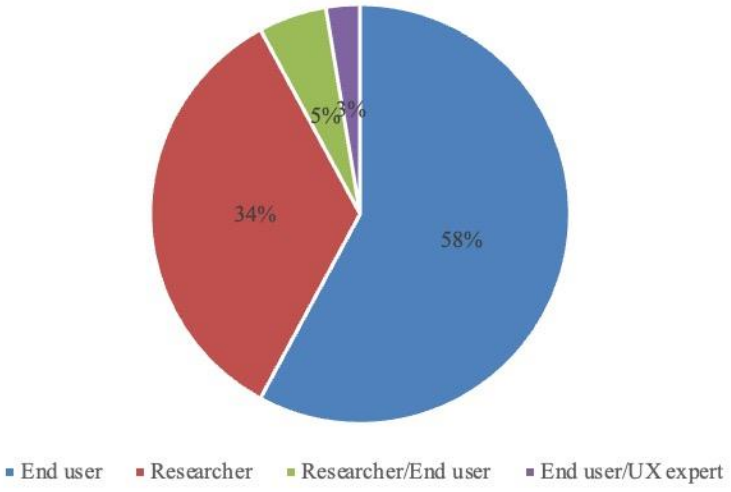


Figure 8: Distribution of the types of people using/studying the tool when doing the evaluations in the papers

2.2.2.4 RQ1-4- What data collection methods are used in publications?

We consider three general types for data collection methods: inspection, inquiry, and testing. Inspection refers to a set of methods that are all based on evaluators feedback about an interface, and includes methods such as heuristic evaluation, cognitive walkthrough, and feature inspection (Nielsen, 1994). This method introduced as a more affordable alternative of usability testing, as it does not require participants (Lewis & Sauro, 2021). Inquiry methods involve user inquiries through interviews or surveys. Testing or user testing methods are a group of methods that are used to gather information from users during their interaction with the product, such as observation, think- aloud, and performance measurements.

Those studies that performed evaluation based on users, employed either the user testing or the inquiry method, and studies performed by researchers usually leveraged the inspection method. There were also some papers that used a combination of them.

As shown in Figure 9, the most-used data collection method is inquiry, which includes conducting surveys or interviews. The inquiry method was used either as the only evaluation method or in combination with other methods. 42% of the papers used the inspection method in their evaluation. Another common approach employed by studies was performing evaluations based on both user testing and inquiry to complement the results, which has shaped 19% of papers. In total, 32% of papers performed some user testing in their data collection methods. More details of data collection instruments will be discussed below.

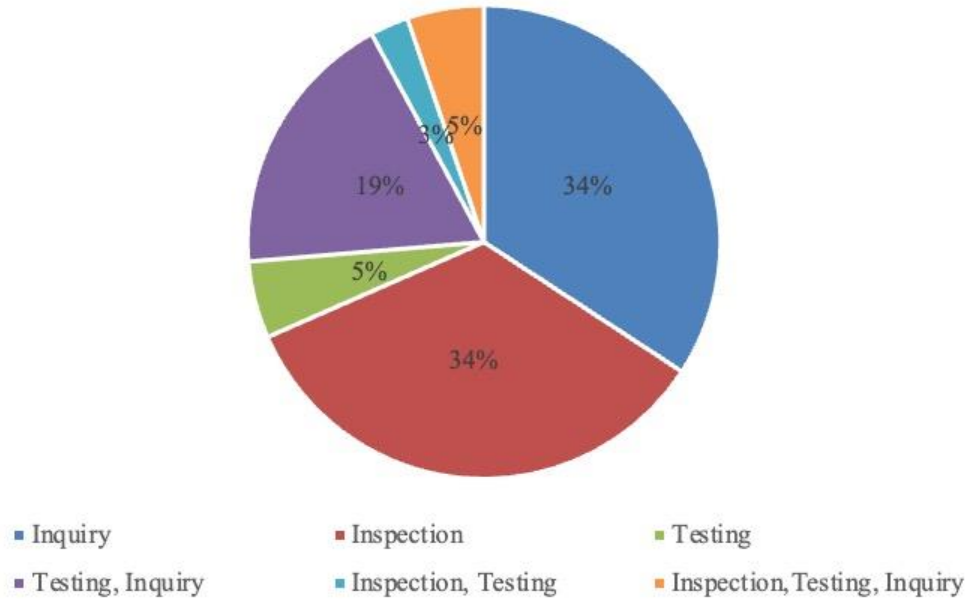


Figure 9: Distribution of general categories of data collection methods employed in the papers

Looking closer at the applied data collection methods and tools, one can see that some papers (30%) benefited from more than one data collection approach. This helps researchers to achieve more reliable results and address more issues and challenges.

The following is a list of data collection tools used by our selected papers that typically involve direct interaction with users:

• **Questionnaires**: This data collection method, which is the most common one in our papers (Agner et al., 2019; Agner & Lethbridge, 2017; Akdur et al., 2018; Badreddin et al., 2018; Lahtinen & Peltonen, 2005; Liebel et al., 2018; Ozkaya & Erata, 2020; Pietron et al., 2018a; Pourali, 2018; Ren et al., 2020; Robles Luna et al., 2018; Safdar et al., 2015; Williams et al., 2014) is one of the most popular methods to gather both quantitative and qualitative user data. It is used either in the pre-evaluation phase to gather demographic information of participants or during the evaluation process for gathering the comments and insights of users about the tool. The SUS (System Usability Scale) (Pietron et al., 2018) and the CSUQ (Computer System Usability Questionnaire) (Pourali, 2018) are two examples applied in the selected papers.

• **Interviews**: This method, which is the second most-used method for user evaluation in our findings (Kuhn et al., 2012; Pietron et al., 2018; Seffah & Rilling, 2001; Stegmaier et al., 2019; Störrle, 2007; Vogelsang et al., 2017; Weber, Zoitl, & HuBmann, 2019; Whittle et al., 2013, 2017) is similar to the questionnaire but allows evaluators to add or change some questions based on the context.

• **Task-based**: This method, as a quantitative approach to usability testing, is performed by giving users tasks and calculating various metrics by researchers (Condori-Fernández et al., 2013; Panach et al., 2011; Planas & Cabot, 2020; Pourali, 2018; Ren et al., 2020; Safdar et al., 2015; Weber, Zoitl, & HuBmann, 2019). The metrics include completion rate and error rate, both for measuring the effectiveness of the tools, as well as elapsed time and number of clicks for measuring the efficiency of the tools. To gather the required information, evaluators usually use screen recording, video recording, eye tracking, or interaction logging. There was one paper (Safdar et al., 2015) that gathered information based on users' self-reports.

• **Think-aloud**: In this method (K.-H. Huang et al., 2011; Pietron et al., 2018; Pourali, 2018; Weber, Zoitl, & HuBmann, 2019), participants are supposed to speak while interacting with a tool and describe what they expect and what happens instead. As a qualitative method, this helps evaluators better understand the users' thoughts and needs. Videotaping, eye tracking (Lewis & Sauro, 2021), audio recording, or just taking notes by researchers are the complementary instruments used for performing this method.

• **User observation**: In this method (Lahtinen & Peltonen, 2005; Stegmaier et al., 2019; Störrle, 2018), evaluators take notes as they watch users interacting with the tool. It could provide some immediate user feedback and is usually used combined with other methods. This method usually utilizes user recording, such as audio recording and videotaping, to enable researchers of interpreting users' behavior after the study. Fernandez (Condori-Fernández et al., 2013) used videotaping to evaluate users' satisfaction based on their facial reactions.

• **Focus group**: This is a qualitative method to explore people's attitudes. It can be used to find out what issues are of most concern for a community or group. This method was only used by one publication (Weber, Zoitl, & HuBmann, 2019), which tried to compare different evaluation methods.

• **Paper prototyping**: This method involves hand-drawn representations of what the product looks like and is usually used in early design iterations of a product development process, which is efficient in cost and time. Huang et al. (K.-H. Huang et al., 2011) employed this method for one of their design iteration processes and stated that, "the paper prototype is not merely an evaluations tool. However, a material for designers to convey their design concepts as well as a platform for communication among domain experts, engineers, and practitioners".

The following are the additional data gathering tools and methods that do not involve direct user interaction and are applied by researchers and authors of the papers:

• **Feature-based evaluation**: This is the most-used method (Egyed, 2006; El Kouhen et al., 2012; Heena & Ranjna, 2011; Mert Ozkaya & Ferhat Erata, 2018; Ozkaya, 2019; Ribeiro et al., 2016; Saraiva & Silva, 2008) among all the methods that did not involve user interaction. This is typically done by rating tools against a list of criteria.

• **KLM-GOMS**: This method predicts how long it will take an expert user to accomplish a routine task without errors using an interactive computer system. It is a useful method to compare the usability of several tools. Two papers (Auer et al., 2007; Planas & Cabot, 2020) applied this method in a comparative study.

• **Usage scenarios**: This is a qualitative method (Ahmar et al., 2015; Ferreira et al., 2014) for early usability evaluation. A usage scenario describes a system's behavior as it responds to requests from an actor who wants to achieve a particular goal, thorough of which usability issues could be highlighted.

• **Heuristic evaluation**: In this method (K.-H. Huang et al., 2011; Weber, Zoitl, & HuBmann, 2019), a group of usability specialists judge and rate products based on a pre-defined set of usability principles.

• **Software analytic**: This method was used by a comparative study (Williams et al., 2014) to evaluate different open-source modeling tools based on their communities.

The bar chart in Figure 10 compares different data collection tools employed by our selected papers, color coded according to whether data is gathered from researcher's own analysis, consultation with external experts, or interaction with tool users. The questionnaire is

the most-used tool in user evaluations; it is one of the most time- and cost-effective techniques. This result is also aligned with (Paz & Pow-Sang, 2016), which presents the questionnaire as the most-used usability evaluation method in software engineering.

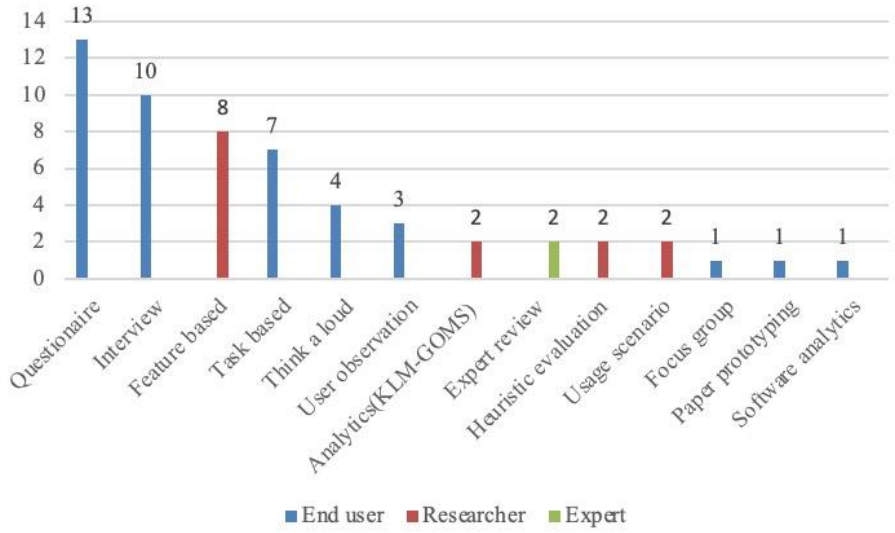


Figure 10: The number of papers in our study that employed each data collection method

2.2.2.5 *RQ1-5- What is the typical number of participants involved in different types of end-user evaluations?*

The number of participants recruited for an evaluation process is another item worth considering, as it could be a vital factor in the result's reliability. We calculated the average number of participants and categorized them based on the papers' data collection approach. Pietron et al. (Pietron et al., 2018), in their proposed evaluation template, recommend at least 9 to 15 participants for performing user testing studies. As we can see in the bar chart, the average number of participants in the selected papers is in the mentioned range.

The results show that papers using questionnaires involve the highest number of participants, with an average of 115. This outcome is not surprising as the questionnaire is one of the most efficient methods in terms of time and cost.

In contrast, the user observation data collection approach has the lowest average number of participants (10). This method needs careful attention to detail while recording users and

analyzing the qualitative data, so it might not always be affordable to perform with a large number of participants.

As illustrated by Figure 11, a boxplot diagram is used to represent the distribution of the number of participants. The distance between the average and median value of the *Questionnaire* method shows some outliers in data. Taking a closer look at those outliers reveals that there are a few studies that conducted remote evaluations (Agner et al., 2019; Agner & Lethbridge, 2017; Badreddin et al., 2018), and therefore contain noticeably more participants than other studies; among which, (Akdur et al., 2018) has the highest amount with 627 participants.

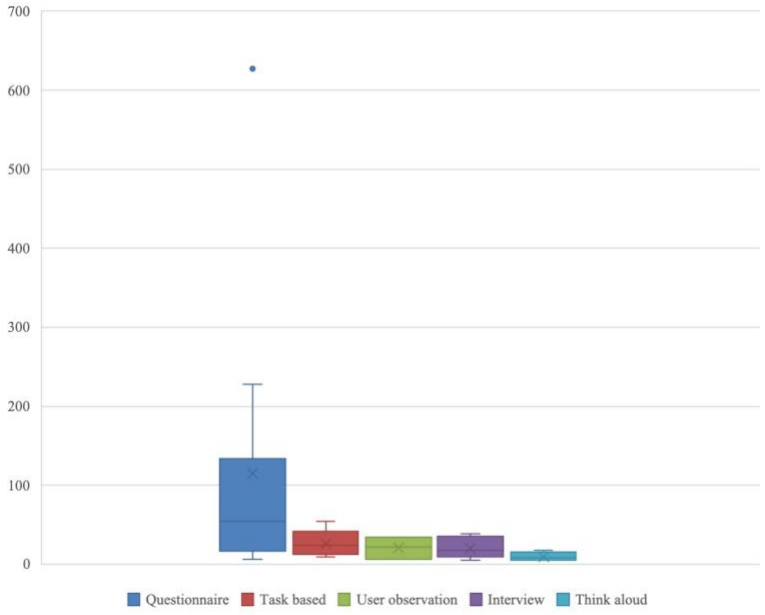


Figure 11: The distribution of the number of participants for each data collection approach in our first SLR

2.2.2.6 RQ1-6- What are the professional profiles of participants in user evaluations?

We analyzed the demographic information from 26 papers that applied tool user evaluation.

Illustrated by Figure 12, in the user testing method, participants should ideally be selected among the real users of the system. Pietron et al. (Pietron et al., 2018) suggest using the TA-EG questionnaire to ensure that the participants have a positive attitude towards technology. The

background of participants depends on the target users of the system. Hence, if a tool is supposed to be used by students, researchers, and also practitioners, the best approach is to recruit participants from both academia and industry.

We identified four groups based on the participants' profiles and categorized the papers based on these groups. The industry group covered 46% of papers, including professionals and practitioners mostly from IT-based companies. 19% of papers performed their evaluation with students (undergraduate and graduate students). 19% of papers did not mention participants' background while stating they recruited participants based on different levels of experience (Badreddin et al., 2018; Condori-Fernández et al., 2013; Panach et al., 2011). One paper claimed using *expert* users without defining the term (Pietron et al., 2018), and one paper (Lahtinen & Peltonen, 2005) used novice users, which they all grouped under *Experience based*. The number of papers involving both students and industry participants was only 11%. There was only one paper (Agner & Lethbridge, 2017) that considered professors who teach modeling as research participants.

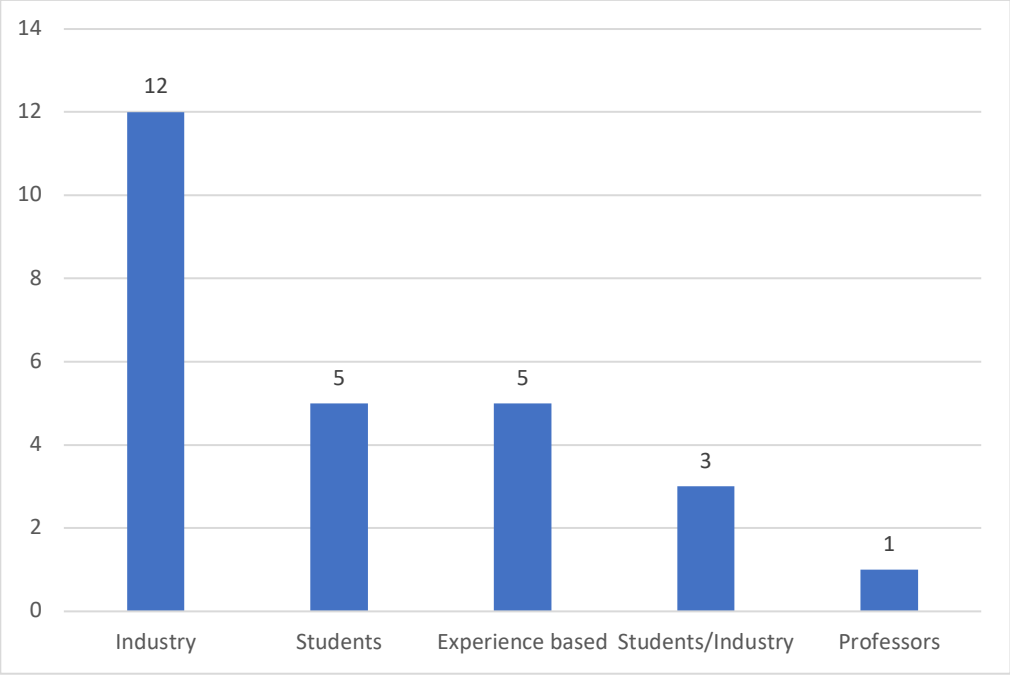


Figure 12: Participants profile in papers

2.2.2.7 RQ1-7- Which tools and model types were most frequently used in the evaluations?

More than 60 modeling tools exist in the market, including commercial and open-source tools (*List of Unified Modeling Language Tools*, 2020). The researcher's preference for choosing the tools to be evaluated is an interesting point to consider. In our analysis, we found almost 50% of papers evaluated just one tool, whereas the remaining evaluated several tools and conducted a comparative analysis.

A variety of tools were highlighted by selected papers, some of which were intentionally selected by researchers to be evaluated, and several tools were mentioned as preferred by users in performed surveys.

In this study we focused on the most-used tools and disregarded those that were mentioned just once. Figure 13 shows tools based on their occurrence in different time-period categories; we used three different time periods, since various tools that were commonly evaluated more than a decade ago are no longer widely used, yet other tools have only appeared recently. As the trend in Fig. 9 displays, Papyrus was the most prominent tool, particularly following 2016, followed by Visual Paradigm, Magic Draw, Enterprise Architect, Eclipse based modeling tools, Argo UML, Umple, Star UML, IBM Rational Rhapsody, and BoUML.

Papyrus is an open-source MBSE tool with a fully customizable environment that can be adopted for various purposes (*Papyrus*, 2020). It is widely recognized as a leading tool by both researchers and practitioners, which explains why it has been utilized more frequently in studies compared to other tools.

Although there are 14 diagrams in UML, only a few of are employed in practice and in evaluation studies. For example, Pietron (Pietron et al., 2018) recommended using state machine diagrams for the study design template, given their familiarity among students and developers. This allows participants from academia and industry to participate in the evaluation.

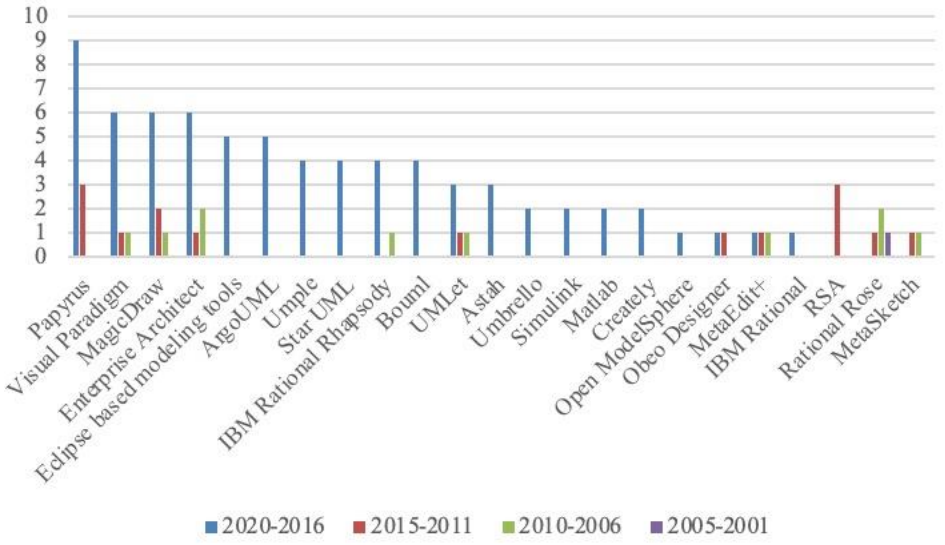


Figure 13: The most-used tools in the studied papers

Figure 14 shows that class diagrams and state machine diagrams are the most popular diagrams used in the evaluations. As the representatives of the system's static and dynamic views, these diagrams can provide a comprehensive view of the system. Following these, sequence diagrams, activity diagrams, and use case diagrams are other diagrams used in the tool evaluation process of our selected papers. As shown by Fig. 10, in total, 12 papers cited particular diagrams in the evaluation process, four of which used more than one diagram, and the remainder used only one diagram. Other papers (29 papers) evaluated programs with no regard for the diagrams.

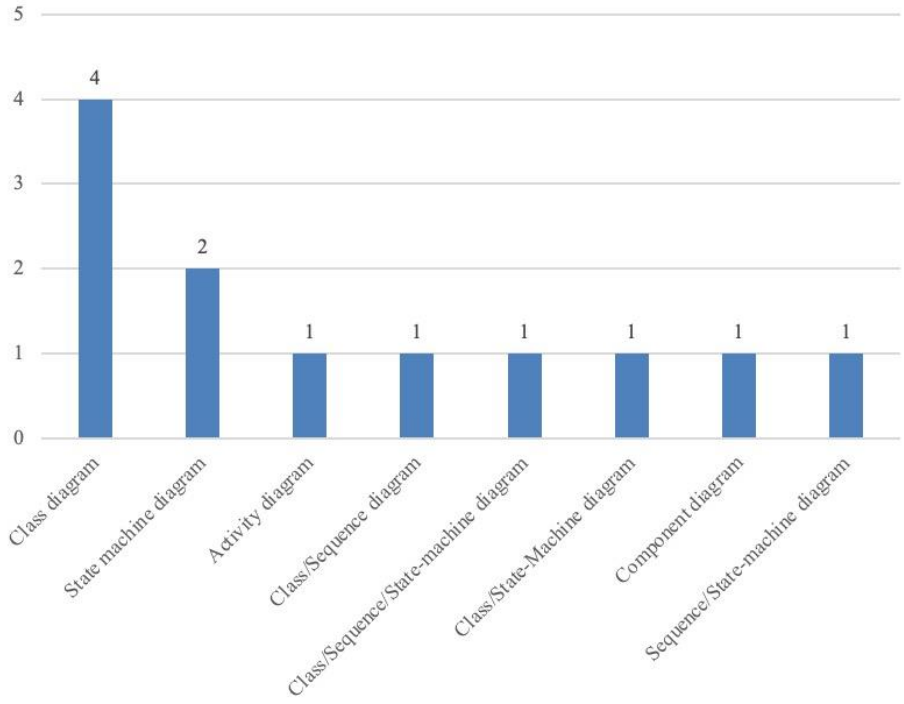


Figure 14: Diagram types used in the analyzed papers

2.2.2.8 RQ1-8- How well are modeling tools evaluated for handling large, complex models?

The ability to construct and maintain large, complex models has been considered a critical feature of modeling tools. However, very few studies addressed this topic.

Störrle (Störrle, 2007) is one of the researchers in this domain who investigated the challenges of large-scale modeling by interviewing some domain experts. There were several significant challenges identified in this study, such as the need for version control and problems related to release and deployment; however, these difficulties are not described in depth. In another study, he presented multi-layer feature to mitigate large models' complexity (Störrle, 2018). Yet, it is validated only from the usability aspect, while problems regarding release and deployment are not considered.

Other papers performed tool evaluation using only simple tasks and small models, and it seems that the real practicality of tools is neglected in evaluations. Therefore, it is imperative to consider this aspect of modeling for further studies.

2.2.2.9 RQ1-9- What are the key challenges of MBE tools identified in publications?

Among all selected papers, 38 papers address particular challenges or issues, either by evaluations or literature reviews.

To gain an overview of existing known challenges, we needed some standard terms and classification since papers have described their identified issues using different terms.

We chose to categorize the issues into the following categories: Utility, Usability, Reliability, Emotional, and Marketing. We arrived at these categories after much discussion in our team with the objective of creating categories that avoid overlap and correspond to the way the issues are discussed in the literature. We attempt to present issues in these distinctive groups, although there are undoubtedly some overlaps.

As (Hartson & Pyla, 2019) defines, Utility "is about the power and functionality of the backend software that gives you the ability to get work (or play) done. It's the real underlying reason for a product or system". To better clarify the gaps, utility issues were further categorized in two subgroups: utility issues in the *tools* being evaluated and utility issues in the *languages* supported by the tools.

According to (Hartson & Pyla, 2019), Usability includes factors such as "ease of use", "user performance", "efficiency", "error avoidance", "learnability", and "retainability (ease of remembering)". Similar to utility, usability issues were further categorized in two subgroups: usability of the tool and usability of the modeling language.

Emotional issues, which are the least considered part of UX in literature, include personal feelings during the usage of system such as fun, enjoyment, pleasure, aesthetics and so on (Hartson & Pyla, 2019).

The Reliability category was used to contain problems related to system bugs.

Marketing issues fit in a group focusing on how easy it is to obtain, install, and update required features of a system. It also considers the cost, availability and quality of the system communities, which could play a critical role in MBSE tools succeeding.

Table 4 summarizes key challenges discussed in the paper reviewed in this SLR, and organizes them into the above categories. The table employs a color-coding system which is aligned with Figure 16: dark blue for usability (tools), orange for utility (language), grey for

usability (tool), yellow for usability (language), light blue for reliability, green for emotional, and navy blue for marketing.

Note that this table only lists the papers that discussed challenges, as indicated in the 'List Challenges' column of Table 2. Note that we will revisit this data in Table 25, incorporating further analysis from later stages of this thesis.

Table 4: MX issues highlighted in each paper of the first SLR

Paper	MX Issues
(Ozkaya & Erata, 2020) Understanding Practitioners' Challenges on Software Modeling: A Survey	**Utility (tool):** Analyzing models, Transforming models, Managing models, Separation of concerns **Utility (language):** Extendibility, Developing formal modeling languages **Usability (language):** Complexity
(Planas & Cabot, 2020) How are UML class diagrams built in practice? A usability study of two UML tools: Magicdraw and Papyrus	**Usability (tool):** Difficult in use, Lack of a proper guidance **Reliability:** Bugginess
(Ren et al., 2020) Collaborative Modelling: Chatbots or On-Line Tools? An Experimental Study	**Utility (tool):** Collaborative modeling feature
(Bucchiarone et al., 2020) Papyrus for gamers, let's play modeling	**Usability (tool):** Complexity
(Ozkaya, 2019) Are the UML modelling tools powerful enough for practitioners? A literature review	**Utility (tool):** Multiple viewpoints, Scalability, Formal verification, Round-trip engineering, Exportation, Collaborative modeling, Scripting, Project management **Usability (tool):** Complexity
(Agner et al., 2019) Student experience with software modeling tools	**Utility (tool):** Feedback, Interaction with other tools **Usability (tool):** Complexity, Slowness, Difficult to use **Reliability:** Bugginess
(Stegmaier et al., 2019) Insights for Improving Diagram Editing Gained from an Empirical Study	**Utility (tool):** Free sketching feature, highlighting feature, template feature **Usability (tool):** User preference in texts, shapes, sizes, colors, Undo functionality **Reliability:** bugginess
(Savary-Leblanc, 2019) Improving MBSE Tools UX with AI-Empowered Software Assistants	**Usability (tool):** Complexity
(Badreddin et al., 2018) A Decade of Software Design and Modeling: A Survey to Uncover Trends of the Practice	**Utility (tool):** Executable artifact, Collaboration and Communication **Usability (tool):** Learning curve, Complexity
(Robles Luna et al., 2018) Challenges for the adoption of model-driven web engineering approaches in industry	**Utility (tool):** Metamodel support, Traceability and Debuggability, Monitoring technologies, Architecture modeling, Technological aspects modeling such as DB connection **Marketing:** Community and tutorials, Expensive licensing, Close source which leads to lack of control,

Paper	MX Issues
(Mert Ozkaya & Ferhat Erata, 2018) Analysing UML Modeling Tools for Practical Use	**Utility (tool):** Various viewpoints, Sub-diagramming for the connectors (e.g., class associations), Formal verification, Collaboration support
(Pourali, 2018) An Empirical Investigation to Understand the Dificulties and Challenges of Sofware Modellers When Using Modelling Tools	**Usability (tool):** Retainability, Error avoidance
(Störrle, 2018) Improving model usability and utility by layered diagrams	**Utility (tool):** Layers feature **Usability (tool):** Complexity
(Agt-Rickauer et al., 2018) DoMoRe – A Recommender System for Domain Modeling	**Usability (tool):** Knowledge intensive
(Akdur et al., 2018) A survey on modeling and model-driven engineering practices in the embedded software industry	**Utility (tool):** Synchronization between software artifacts, Version management, Integration of legacy code, Model checking, Traceability and compatibility, Code generation **Usability (tool):** Need training
(Liebel et al., 2018) Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice	**Utility (tool):** Interoperability between MBE tools, Variability management, Version management **Usability (tool):** Need training
(Agner & Lethbridge, 2017) A Survey of Tool Use in Modeling Education	**Utility (tool):** Interaction with other tools, Supporting modeling aspects **Usability (tool):** Complexity
(Whittle et al., 2017) A taxonomy of tool-related issues affecting the adoption of model-driven engineering	**Utility (tool):** Model analysis, Scalability, Refactoring models, Sketching models, Tool versioning, Flexibility, Customizability, Integration, Migration **Usability (tool):** Complexity, Not match human mental model, Productivity **Usability (language):** Complexity **Emotional:** Trust **Marketing:** Cost of tools, Sustainability
(Vogelsang et al., 2017) Should I Stay or Should I Go? On Forces that Drive and Prevent MBSE Adoption in the Embedded Systems Industry	**Utility (tool):** Incompatibility, Traceability, Modularization, Testing **Usability (tool):** Learnability, Complexity **Marketing:** ROI uncertainty
(Bordeleau et al., 2017) Challenges and Research Directions for Successfully Applying MBE Tools in Practice	**Utility (tool):** Collaboration facilities and customization, Graphical and textual support in a tool, Model diff/merge, Model review, Document generation. **Usability (tool):** Usability issues
(Ribeiro et al., 2016) Comparative analysis of workbenches to support DSMLs: Discussion with non-trivial Model-Driven Development needs	**Utility (tool):** Validation Support, Transformation support **Usability (tool):** Learnability issues
(Safdar et al., 2015) Empirical Evaluation of UML Modeling Tools–A Controlled Experiment	**Usability (tool):** Learnability issues, Productivity issues
(Ahmar et al., 2015)	**Usability (tool):** Readability issues

Paper	MX Issues
Enhancing the communication value of UML models with graphical layers	
(Ferreira et al., 2014) Characterizing the Tool-notation-people Triplet in Software Modeling Tasks	**Utility (tool):** Visibility, Partial semantic verification **Usability (tool):** Hard mental operation **Reliability:** Error proneness
(Williams et al., 2014) Software Analytics for MDE Communities	**Marketing:** Lack of rich community
(Whittle et al., 2013) Industrial Adoption of Model-Based Engineering: Are the Tools Really the Problem?	**Usability (tool):** HCI concerns **Marketing:** Lack of communities
(Condori-Fernández et al., 2013) An empirical approach for evaluating the usability of model-driven tools	**Utility (tool):** Customizability **Usability (tool):** Novice users are not guided, Interfaces are not visually consistent, Some interfaces provide too much information with regard to the available space, Error messages do not help the user to solve the mistake, Icons are not self-explicative, Some interface titles are confusing, The tool does not provide undo and redo facilities, Some elements do not provide a menu when the user clicks with the right button of the mouse, Some interfaces are obtrusive (they do not allow showing the window below), Some functions are only reachable by means of icons, but not through the menu
(Kuhn et al., 2012) An Exploratory Study of Forces and Frictions Affecting Large-Scale Model-Driven Development	**Utility (tool):** Model diffing, Point-to-point traceability, **Utility (language):** Lack of problem-specific visual "little languages" **Usability (tool):** Long build-cycles prevent live modeling
(Heena & Ranjna, 2011) A comparative study of UML tools	**Utility (tool):** Round-trip engineering, All UML diagrams, UML 2.0, Document generation
(Panach et al., 2011) An Experimental Usability Evaluation Framework for Model-Driven Tools	**Utility (tool):** Compatibility issues **Usability (tool):** Guidance, Error management, Consistency and Adaptability violations, User control
(Huang et al., 2011) Hammering Models: Designing Usable Modeling Tools	**Utility (tool):** Shortcomings in terms of supported functionality **Usability (tool):** Not support different levels of expertise
(Eichelberger et al., 2009) A Comprehensive Survey of UML Compliance in Current Modelling Tools	**Utility (tool):** Compatibility problems
(Saraiva & Silva, 2008) Evaluation of MDE Tools from a Metamodeling Perspective	**Utility (tool):** Model transformations **Utility (language):** Support for language specification (syntax and semantics) **Usability (tool):** Complexity
(Auer et al., 2007)	**Usability (tool):** Complexity

Paper	MX Issues
Explorative UML modeling comparing the usability of UML tools	
(Störrle, 2007) Large Scale Modeling Efforts: A Survey on Challenges and Best Practices	**Utility (tool):** Version control, Modularity, Migration, Releases, Backups and deployment
(Egyed, 2006) Instant consistency checking for the UML	**Utility (tool):** Consistency checking
(Lahtinen & Peltonen, 2005) Adding speech recognition support to UML tools	**Usability (tool):** Complexity
(Seffah & Rilling, 2001) Investigating the relationship between usability and conceptual gaps for human-centric CASE tools	**Usability (tool):** Interface personalization, Retainability, Error avoidance, Speaking the developer's language, Keeping the developer informed the IDE status

After reviewing and categorizing all the issues, we counted items in each group based on their occurrence. If a paper points to one or several items in a specific group, it still counts as one occurrence for this group.

The hierarchy and proportion by each category and subcategory of issues, is illustrated in Figure 15, highlighting that most repeated issues by papers are in the "Utility" and "Usability" groups, with a focus on tool issues, whereas "Emotional" and "Reliability" are the groups with the fewest reported issues.

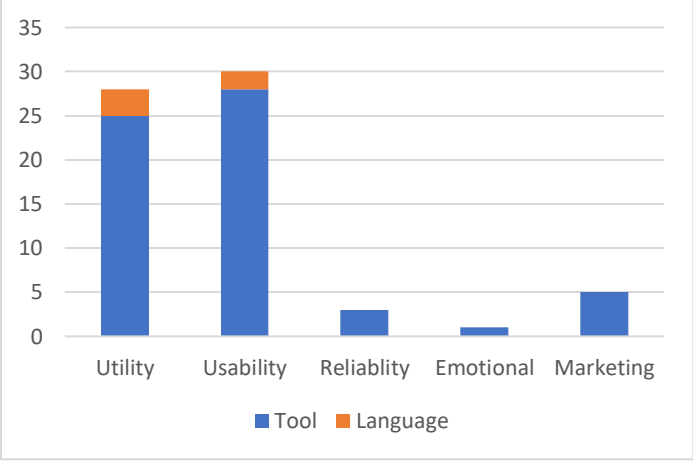


Figure 15: Issue categorization in the MX literature

Figure 16, illustrates the hierarchy and proportion by each category and subcategory of issues, showing that more than half of issues were concerned with tools' utility.

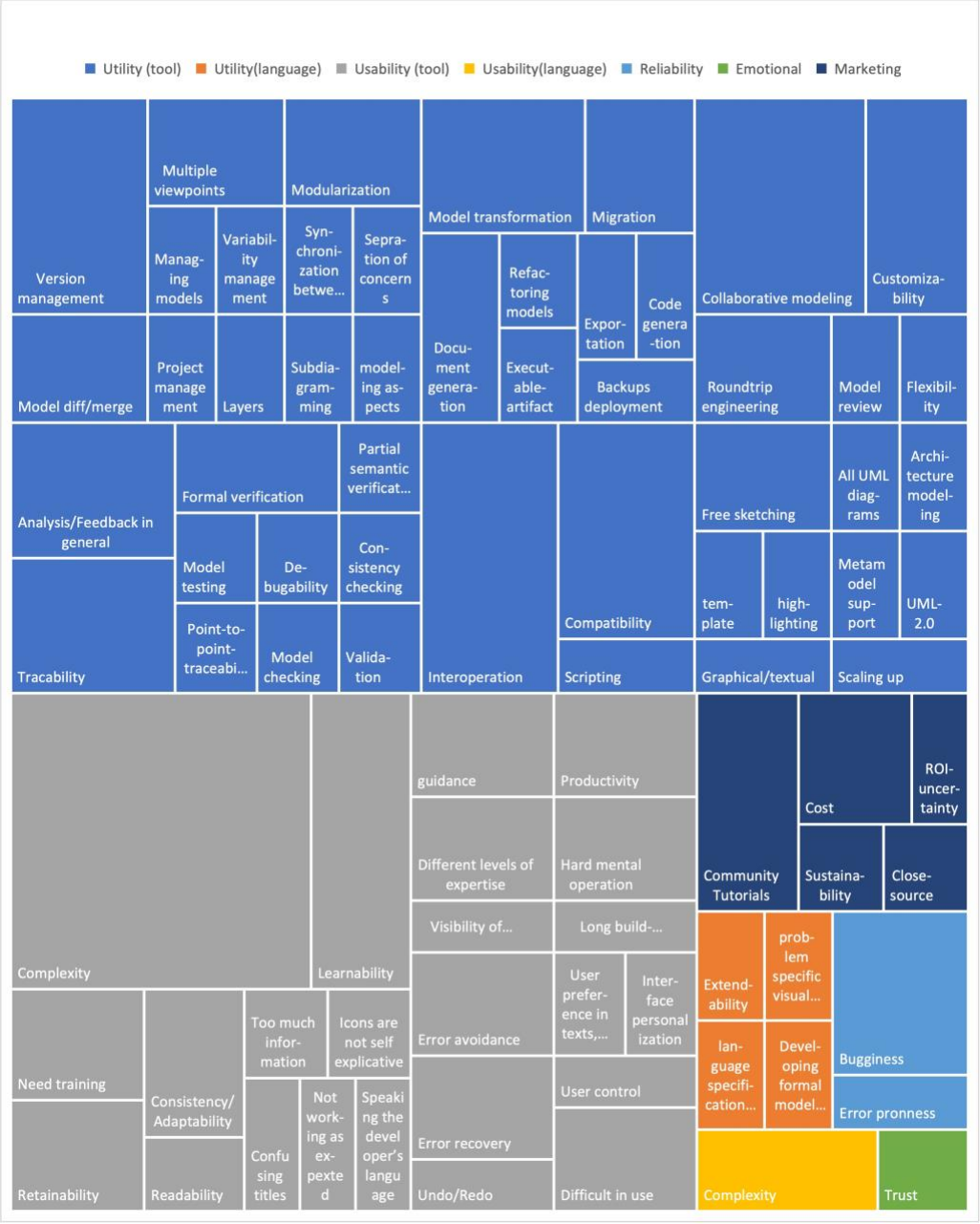


Figure 16: Issue category proportions from our first SLR as a tree map

2.2.3 *Discussion and Implications From the first SLR*

This study has presented a systematic literature review of usability evaluation methods and has identified challenges for software modeling tools. Snowballing was applied to finding

papers starting from a seed set that was determined from a search. Overall, 41 papers were selected to review. Extracted data were categorized and described based on several research questions, through which various limitations and research gaps were identified. These are discussed below.

The following are research gaps that we recommend be considered by future researchers who are studying, improving, or comparing specific MX qualities:

Benefit from the interview method in research design: Our results have shown that the interview method provides challenges in a wider variety of categories than other methods. 50% of papers that present the most issues (from three or more issue categories) have utilized interviews in their research design. The interview method is even more significant in the UX-focused category than the utility category. We adopt this method in this thesis.

Distribute testing tasks based on user profiles: In terms of sample selection in the user-testing evaluation method, representing the full spectrum of targeted users is important. Our findings indicate that the majority of papers recruited their participants from either academia or industry. Future researchers need to consider other aspects of user distribution, including by level of experience in the domain and expertise in using the system. As Lewis and Sauro say, "To determine who will participate in the test, the administrator needs to obtain or develop a user profile" (Lewis & Sauro, 2021). It is important for evaluators to establish a user profile and its characteristics to increase the representativeness of participants, and hence the validity of studies. Later in this thesis we categorize users by expertise in modeling and experience in industry.

Involve UX experts in analysis: Based on our results, only one paper (Weber, Zoitl, & HuBmann, 2019) benefited from the input of a UX expert. Our analysis also shows that the papers have the fewest improvement suggestions in the "Human factors" group. Consequently, we require more empirical studies in the evaluation of MBSE tools that incorporate UX expert feedback, alongside user studies, in order to gain an overall view and provide reliable suggestions for improving the user experience. In this thesis work, we benefited from having a UX expert (the supervisor) involved.

Test for scalability using large models: Many software systems are very large, yet the literature focuses on evaluation with small models, there is a need for more studies regarding

scalability as a critical feature for modeling tool practically. In this thesis we interviewed people who do work on large models.

Assess MX in areas other than just usability and utility: According to the selected papers, there has been a gap in studies concerning emotional factors and marketing practices, which highlights the necessity to carry out more empirical research in this area. In recent years, marketing issues have received increased attention, as app stores, ease of installation and updatability have become essential factors influencing awareness and popularity. More studies are required to identify all the impactful factors of these types. We address this in the studies we undertake later in this thesis.

Consider collaborative modeling as an important factor contributing to MX: While we categorized collaborative modeling issues as a utility category, it is worth noting that it has a broader effect, as it could impact user performance (usability) as well as user enjoyment (emotional). To facilitate effective team collaboration, modeling tools should provide good support for versioning, model diff/merge, model review, and document generation (Bordeleau et al., 2017). Therefore, a significant portion of utility issues require improvement to contribute to collaborative modeling support.

Consider language issues as well as tool issues: In most cases, the reported issues are related to the modeling tools used, but language quality plays an equally important role in the overall user experience. Not differentiating between language and tool issues, could cause misleading results in the evaluation process. Considering this, Ferreira (Ferreira et al., 2014) provided a conceptual tool to address tools issues using tool-notation-people triplets, which covers all kinds of issues. That paper suggested evaluation methods in each category but does not state any sub-category or criteria for using it as a guideline for tool evaluations.

The SLR reported in this section has tried to provide data regarding these deficiencies by providing detailed categories and sub-categories in MDE UX evaluation. Achieving a comprehensive set of guidelines that consider both tool and language factors in MX, should be a goal.

2.3 Systematic Literature Review on Human-Centric Barriers to the Adoption of Software Modeling

In the previous section, we discussed our primary literature review performed in 2020, where we analyzed and evaluated existing research addressing challenges hindering the widespread adoption of software modeling tools. In Chapter 5 we will discuss the grounded theory analysis we performed, in which we revealed that the predominant barriers faced by practitioners are not primarily technical or tool-specific; rather, they are deeply rooted in social and organizational factors.

Consequently, recognizing this gap, in 2024, we conducted another literature review to deepen our understanding with a focus on non-technical issues, and to extract the issue classifications reported by studies. The following subsections will outline the key methodologies employed and highlight the main findings in the second SLR.

The content of this section was developed subsequently to our grounded theory work in Chapter 5, so one option was to place this literature review in that chapter or in a separate subsequent chapter. However, we believe that from the perspective of the reader of this thesis, it makes more sense to read about all the background literature in one place.

2.3.1 Methodology of the SLR of human-centric issues

For the current literature review, we utilized the same methodology as in our previous study, employing snowballing guided by the principles outlined by Wohlin (Wohlin, 2014). Employing both forward and backward snowballing techniques, we continued the search process until saturation was achieved, and no further novel papers were uncovered.

The research questions for this new SLR were:

RQ2- What are the human-centric barriers to the adoption of software modeling identified in the literature?

RQ3- What types of classifications of adoption barriers are reported in the literature?

2.3.1.1 Search Process

We used Google Scholar to identify the start set and initiated the search process by using the following expressions: “Challenges or issues in X”, where X is one of following three strings: “MDE or Model-Driven Engineering”, “MBSE or Model-Based Software Engineering”, “Software Modeling”.

We ran the queries separately in Google scholar and extracted the first 30 papers ordered by the search algorithm’s computation of relevance. These were the papers that appeared on the first three pages of Google Scholar.

We manually screened the papers, following the same process used in our previous literature review.

Following the above screening, we obtained a primary set of nine most-relevant papers, which we have marked with an asterisk (*) in Table 6. Backward and forward snowballing in the first iteration gave 1242 results, of which 6 papers were selected (using the same manual method as used in previous literature review) to include in our pool. In the second iteration, we found 530 papers, among which 2 papers were included. For the last iteration, from 125 papers, we could not find any new relevant papers for our study. Therefore, the snowballing procedure for this study was finished by finding 17 papers for further analysis, listed in Table 6.

Three of these 17 papers (Akdur et al., 2018; Vogelsang et al., 2017; Whittle et al., 2017) had also been reviewed in our previous SLR, meaning that they were relevant to both tool issues and non-tool issues. There papers are marked with an $ in Table 6.

2.3.1.2 Inclusion and Exclusion Criteria

We selected references based on the inclusion and exclusion criteria stated in Table 5. The criteria are constructed based on the research goal and question.

Table 5: Inclusion and exclusion criteria for our SLR regarding non-tool modeling issues

Inclusion Criteria	Exclusion Criteria
Papers in the English language.	Papers that only focus on technical tool-related issues.
Papers published between the years 2014-2024 inclusive.	Papers that only focus on modeling languages.
Papers that highlight modeling adoption challenges.	Grey literature (theses, non-refereed papers, manuals, books)
Peer-reviewed papers only.	

2.3.1.3 Data Analysis Process for the second SLR

We employed a qualitative method for data analysis, following the same process used in our previous literature review.

In our coding process, we implemented directed coding to align our findings with our empirical findings reported in Chapter 5.

2.3.2 Results of the second SLR regarding human-centric issues in modeling adoption

The list of selected papers based on their publication year, publication type, authors and titles are shown in Table 6.

During this SLR, our search resulted in 3 papers – that were also retrieved as in our earlier SLR (Section 2.2.2.9). For these papers, we aimed to explore human-centric challenges only, although some overlap with the issues reported in Section 2.2.2.9 may have occurred. To maintain accuracy in our analysis, we will avoid double counting these issues when revisiting the literature in Section 7.1.

Table 6: List of selected papers in the SLR2 regarding non-technical issues (* shows the first set of papers | $ shows repeated papers from SLR1|

Ref.	Title	Conf./ Journal	Year
(Holtmann et al., 2024)	Processes, methods, and tools in model-based engineering—A qualitative multiple-case study	J	2024
(Henderson & Salado, 2024)	The effects of organizational structure on MBSE adoption in industry: Insights from practitioners	J	2024
(Henderson et al., 2024)	MBSE adoption experiences in organizations: Lessons learned	J	2024
*(Verbruggen & Snoeck, 2023)	Practitioners' experiences with model-driven engineering: a meta-review	J	2023
(Escalona et al., 2022)	A quantitative SWOT-TOWS analysis for the adoption of model-based software engineering	J	2022
(Akundi et al., 2022)	Perceptions and the extent of model-based systems engineering (MBSE) use – An industry survey	C	2022
*(Bucchiarone, Cabot, et al., 2020)	Grand challenges in model-driven engineering: an analysis of the state of the research	J	2020
(Huldt & Stenius, 2019)	State-of-practice survey of model-based systems engineering	J	2019
*(Chami & Bruel, 2018)	A survey on MBSE adoption challenges	C	2018
$*(Akdur et al., 2018)	A survey on modeling and model-driven engineering practices in the embedded software industry	J	2018
*(Giraldo et al., 2018)	Considerations about quality in model-driven engineering current state and challenges	J	2018
*(Abrahão et al., 2017)	User experience for model-driven engineering: challenges and future directions	C	2017
$*(Vogelsang et al., 2017)	Should I stay or should I go? On forces that drive and prevent MBSE adoption in the embedded systems industry	C	2017
$*(Whittle et al., 2017)	A taxonomy of tool-related issues affecting the adoption of model-driven engineering	J	2017
(Morris et al., 2016)	Issues in conceptual design and MBSE successes: Insights from the model-based conceptual design surveys	J	2016
(Bonnet et al., 2015)	Implementing the MBSE cultural change: Organization, coaching and lessons learned	J	2015
*(Vallecillo, 2015)	On the industrial adoption of model driven engineering. Is your company ready for MDE?	J	2015

2.3.3 *What are the human-centric barriers to the adoption of software modeling identified in the literature?*

The existing literature highlights numerous non-technical and non-tool-related challenges that hinder the adoption of software modeling practices. These challenges encompass a wide array of factors, ranging from organizational barriers to individual perceptions and external pressures.

To simplify comprehension, we structured the extracted factors from the literature into the same general themes that we will also use in Section 5.4. It is important to note that not all themes emerged from every method used in this thesis. These discrepancies will be addressed and discussed in detail in Chapter 7.

It should be noted that many of these issues are common to the adoption of *any* new software or processes, such as compilers, project management tools, testing processes, and so on.

2.3.3.1 *Theme A. Tooling and Support Concern*

While our second SLR's primarily objective was to address concerns beyond tooling and technical matters, it is worth noting that delineating a clear boundary between these two groups is challenging. So, the following are the factors extracted from literature.

A primary concern has consistently been the **limited utility of tools**, especially in terms of their compatibility and integration with existing systems. As stated by Vogelsang et al. (Vogelsang et al., 2017). "Incompatibility of MBSE tools with existing tools is a specific inertia force that prevents MBSE adoption." This is highlighted by other researchers who have noted issues such as "lack of flexibility" (Henderson & Salado, 2024) and "lack of integration with development frameworks" (Escalona et al., 2022).

Obsolescence is another issue, as customers worry about being locked into specific tools, raising concerns about long-term data portability (Holtmann et al., 2024). **Security and confidentiality** are also major concerns, with potential risks associated with data security in these tools being highlighted by several studies (Bucchiarone, Cabot, et al., 2020). Additionally,

(Bonnet et al., 2015) discuss how some organizations implement IT policies that prohibit the use of software modeling tools due to security concerns.

The literature reveals that model-driven engineering (MDE) tools often lack a focus on non-technical users. This imbalance suggests that the development of MDE tools often overlooks the **usability requirements** of a broader range of users, particularly those without a technical background (Bucchiarone, Cabot, et al., 2020). According to Abrahão (Abrahão et al., 2017) "MDE tools were largely oriented to engineers, rather than considering diverse types of users." Another paper (Vallecillo, 2015) highlighted this sentiment, stating, "There is too much emphasis on technology and not enough on technology users and their needs."

Modeling tools and processes often introduce a layer of **complexity** that hinders their adoption due to fears of a long learning curve. Several studies have noted this issue, highlighting the significant effort and **cost** required to overcome it (Abrahão et al., 2017; Akundi et al., 2022; Bonnet et al., 2015; Huldt & Stenius, 2019; Verbruggen & Snoeck, 2023; Vogelsang et al., 2017).

Additionally, there is a noted **deficiency in the support** provided by tool vendors and the modeling community. Bucchiarone et al. (2020) identify the need for better facilities to store and access modeling artifacts: "The adoption of such repositories is sporadic in the community". This topic was also brought to light by other researchers (Escalona et al., 2022; Whittle et al., 2017), where they mentioned 'Open Community' as a category of tool-related adoption issues. This category focuses on the presence or absence of an open community where tool users can receive peer support and examines whether developer forums exist to offer examples, advice, and assistance to users.

There is also a notable inadequacy or flaw in the information available about MDE concepts, goals, tools, achievements, and best-practice examples, which creates barriers for those seeking to understand and adopt these technologies effectively (Morris et al., 2016; Vallecillo, 2015).

2.3.3.2 Theme B. Challenges from Modern Work Paradigms

Over the last two decades, agility has come to dominate many software engineering projects. Agility promotes early and frequent iterations, each providing incremental value, and reducing risk by not attempting to deliver a large working product after many months that may have incorporated poor requirements. Agility is supported by techniques such as constant re-evaluation of priorities, interaction with users, automated testing, careful version control, and continuous integration. Most of its proponents do not claim it eliminates design and documentation, but its practitioners tend to see delivering working code as key. Agile practices have been further enhanced by the use of many modern software frameworks that provide sophisticated services to applications and reduce the volume of code that needs to be written manually.

There is thus controversy as to whether agility and modeling can or should co-exist; some research suggests that model-based engineering (MBE) and **agile methodologies** can be compatible and even mutually beneficial (Holtmann et al., 2024), while other papers argue that there are fundamental contradictions between the two (Bucchiarone, Cabot, et al., 2020). Proponents of their compatibility highlight that MBE can enhance agility by enabling faster changes and providing higher abstraction levels in development artifacts. They argue that with the right tools and practices, MBE can seamlessly integrate into agile workflows, supporting iterative development and rapid adaptation to changes. On the other hand, critics point out that MBE's emphasis on detailed upfront modeling may conflict with agile methods' focus on flexibility and minimal initial documentation. They assert that the rigid structures often associated with MBE could impede the dynamic and evolving nature of agile projects. Thus, while there is evidence supporting both perspectives, the successful integration of MBE and agility requires careful consideration of the specific context, tools, and practices used.

2.3.3.3 Theme C. Behavioral Resistance

Resistance to adopting MBE is a significant issue in the literature, often exacerbated by a **lack of top management support**. Without backing from senior leadership, initiatives to implement new technologies struggle to gain traction (Akundi et al., 2022; Chami & Bruel,

2018; Henderson et al., 2024; Huldt & Stenius, 2019; Vogelsang et al., 2017). This lack of support is evident in various ways, including insufficient allocation of resources, which increases pressure on employees (Holtmann et al., 2024). Cultural factors, such as a conservative mindset and the prevalence of coding culture, further contribute to this resistance (Vallecillo, 2015).

Developer resistance is a significant factor highlighted by multiple studies and is closely linked to the lack of a change management strategy. This resistance can be attributed to various factors:

- Lack of Awareness and Expertise: Chami and Bruel (Chami & Bruel, 2018) identify a lack of awareness and expertise as key contributors to resistance.
- Lack of Organizational Support: Giraldo et al. (Giraldo et al., 2018) point out that insufficient organizational support for MDE adoption exacerbates resistance. Bonnet et al. (Bonnet et al., 2015) further emphasize that the absence of cultural change and the fear of change within the organization are significant contributors to this resistance.
- Personal Reluctance: Whittle et al. (Whittle et al., 2017) note that personal reluctance to try new things and a lack of cultural change within the organization contribute to fear of change.
- Conservative Mindset: Vallecillo (Vallecillo, 2015) blames the conservative mindset of many software practitioners for this inertia.

Holtmann et al. (Holtmann et al., 2024) in their interesting study, provide a detailed analysis of various types of resistance:

- Resistance to Top-Down Change: Engineers often resist changes imposed by management, especially when excluded from decision-making processes.
- Resistance to External Change: New trends, tools, or standards introduced from outside can clash with established company cultures and practices.
- Inter-Company and Inter-Unit Resistance: Changes introduced by collaborators or different units within the same company often face resistance due to misalignment in responsibilities and ways of working.

- Resistance from Standardization Bodies: Standardization bodies are often slow to accept models as evidence for compliance, preferring traditional textual documentation.
- Personal Resistance: Increased workload, the need to acquire new skills, conflicting performance indicators, and personal factors such as nearing retirement can all contribute to individual resistance.

Additionally, Escalona et al. (Escalona et al., 2022) suggest that the coding culture prevalent among developers is a significant factor preventing change.

The **lack of a clear strategy for change management** also plays a crucial role in this resistance. Without a defined purpose and strategy, MBE adoption is likely to fail. Studies emphasize the importance of aligning MDE with existing organizational processes and adapting to the specific context (Bonnet et al., 2015; Chami & Bruel, 2018), "a crucial basis for MBSE adoption is to define a clear purpose and scope (the why and what). Ideally, it must be precisely described before beginning the deployment." Therefore lacking a clear purpose and modeling objective could lead to adoption failure (Bonnet et al., 2015).

There are different approaches to MBSE adoption: "off-cycle (in a sandbox environment) or on-cycle (directly on productive projects) (Chami & Bruel, 2018).", Selecting the right strategy for the specific organizational context is crucial (Henderson et al., 2024). This concept, referred to as "Context Inertia" by Vogelsang et al. (Vogelsang et al., 2017), describes the challenge when the current development process does not align with MBSE techniques. Both legacy systems and the need to change developers' mindsets and workflows can become significant obstacles to successful modeling practices.

Giraldo et al. (Giraldo et al., 2018) stress the importance of aligning and adapting model-driven engineering (MDE) with the existing ways people and organizations operate. Similarly, Whittle et al. (Whittle et al., 2017) highlight the necessity of understanding the extent of process change required by the introduction of a new tool.

In a study on practitioners' experiences with model-driven engineering, Verbruggen and Snoeck (Verbruggen & Snoeck, 2023) point to the lack of method maturity and an organization's culture as inhibitors of the adoption of modeling practices.

Organizational structure as part of the change management plan should be considered when adopting modeling. Recent studies have demonstrated a correlation between organizational factors – such as size, formalization, centralization, and others – and the success or failure of modeling adoption (Henderson et al., 2024; Henderson & Salado, 2024).

These factors underline the importance of a well-planned and context-sensitive change management strategy to ensure successful adoption and integration of MBSE.

2.3.3.4 Theme D. Lack of Perceived Value

A recurring issue in the literature is the lack of perceived value in using MDE tools. a paper examining the current state of practice in MBSE highlights that a "lack of perceived value" hinders the effective introduction of a model-based approach (Huldt & Stenius, 2019).

Some companies do **not see pressing problems** that MBE could solve, leading to a lack of motivation to adopt these tools (Holtmann et al., 2024).

Negative past experiences with immature tools that have poor user experience, low stability, and lack basic features, further discourage practitioners from trying modeling again, as reported by (Verbruggen & Snoeck, 2023; Vogelsang et al., 2017).

Additionally, the uncertainty surrounding the return on investment (ROI) in MBSE implementation presents a significant challenge. The high initial costs and difficulty in justifying these investments contribute to this uncertainty (Chami & Bruel, 2018; Henderson et al., 2024; Verbruggen & Snoeck, 2023; Vogelsang et al., 2017; Whittle et al., 2017).

This investment mostly comes from the high cost of software modeling tools, However, it may also arise from the training expenditures for employees, which further contribute to the overall uncertainty and apprehension regarding the financial implications of MBE adoption (Giraldo et al., 2018).

It is important to note that the issue of cost becomes more daunting when it cannot be justified within the company. As highlighted in various studies, challenges such as difficulty in quantifying investments (Escalona et al., 2022; Vallecillo, 2015) and therefore justifying costs (Akundi et al., 2022) hinder the adoption of MBSE. Vogelsang et al. (Vogelsang et al., 2017) discuss the uncertainty surrounding return on investment (ROI) in MBSE implementation,

indicating the complexities and risks involved in estimating the benefits. Huldt and Stenius (Huldt & Stenius, 2019) emphasize the risks associated with the adoption of MBE, further complicating the assessment of ROI.

2.3.3.5 Theme E. Knowledge and Skill Deficiency

Deficiency in knowledge and skills is a significant barrier to the successful adoption of new methodologies like MBE. Many organizations lack adequately trained personnel, and the cost and effort required to train employees are significant hurdles. This expertise gap can lead to improper implementation and suboptimal results, further complicating the adoption of MDE. Notably, 10 out of 17 studies have identified the lack of employee knowledge and skills in modeling as a major hurdle in adopting MBSE, highlighting the importance of this factor (Abrahão et al., 2017; Akdur et al., 2018; Akundi et al., 2022; Escalona et al., 2022; Henderson et al., 2024; Holtmann et al., 2024; Huldt & Stenius, 2019; Vallecillo, 2015; Whittle et al., 2017).

2.3.4 What types of classifications of adoption barriers are reported in the literature?

Studies in the literature have reported issues in MBE adoption in two ways: one lists various challenges and issues, while the other classifies these issues into specific themes and sub-themes. We are particularly interested in the latter and explore and extract these classifications in this section. After reviewing all papers, we found that 5 out of 17 papers are in the second group. Below is a review of the classification proposed by these papers.

1. Henderson et al. (Henderson et al., 2024), in their MBSE adoption lessons learned categories classified issues into three different categories:
    - *Organizational design* with factors: workforce knowledge/skills, integration, demonstrated benefits/results, and organizational structure.

- *Organizational enablers/barriers* with factors: leadership/management/support/commitment, training, resources for implementation, tool infrastructure.
- *Organizational change management* with factors: application of MBSE methods/processes and modeling practices, adoption/implementation strategy and design, culture change management, willingness to use tools.

2. Akundi et al. (Akundi et al., 2022) have classified MBSE adoption challenges into 4 themes: tools related, knowledge related, cultural/political/cost related, customer related.
3. Bucchiarone et al. (Bucchiarone, Cabot, et al., 2020) proposed a classification of challenges encompassing foundational, domain-specific, tool-related, community, and social factors. Figure 17 illustrates these factors and their sub-factors as identified in their study.

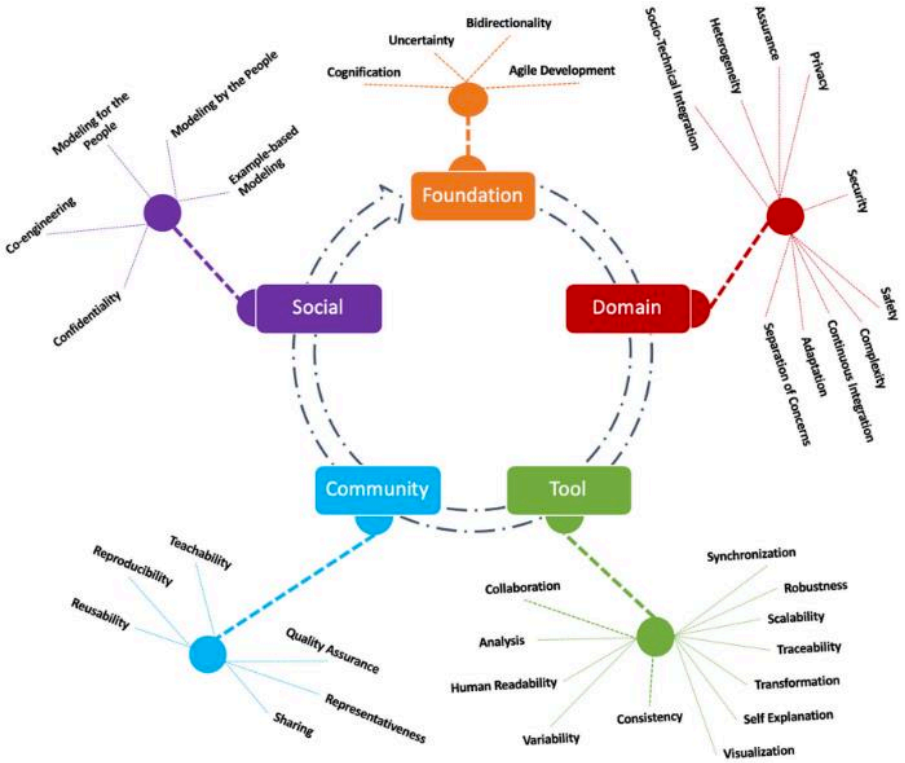


Figure 17: Challenges classification (Bucchiarone, Cabot, et al., 2020)

4. Whittle et al. (Whittle et al., 2017) in their study proposed a taxonomy of tool-related issues affecting the adoption of model-driven engineering. This taxonomy includes four general themes, which are further divided into categories and sub-categories. Below is a summarized list of these categories and sub-categories.
    - *Technical categories* include tool features, practical applicability, complexity, human factors, theory, impact on development.
    - *Internal Organizational categories* include processes, organizational culture, skills.
    - *External organizational categories* include external influencers, commercial aspects.
    - *Social categories* include control, and trust.

5. Valecillo (Vallecillo, 2015) examines the current state of MDE in industry and categorizes the obstacles for adoption into three main areas: technical, cultural, and economic, providing examples for each. Technical challenges include immature tools, undefined semantics, lack of documented processes, absence of model validation tools, and difficulty in choosing the right MDE variant. Cultural obstacles involve a lack of education and experience, an overemphasis on technology over user needs, insufficient information about MDE, resistance to change, and a lack of abstraction skills. Economically, companies fear excessive costs, short-term business focuses that discourage investment, and the challenge of quantifying upfront investments. These combined factors make many practitioners view MDE as unproven or a fading trend, hindering its widespread adoption in the industry.

2.4 Threats to Validity

The systematic literature reviews present several potential threats to validity that may influence their outcomes and conclusions. We address each category of validity and the steps taken to mitigate these threats.

- **Construct Validity**

  Inadequate Search Terms and Criteria: The use of inappropriate or incomplete search terms and inclusion/exclusion criteria could result in missing relevant studies or including irrelevant ones. ***Mitigation****: We applied a well-defined snowballing approach, reviewed by external peers, to enhance the comprehensiveness of our search. Additionally, the study protocol was peer-reviewed and consistently followed.*

- **Internal Validity**

  Bias in Study Selection and Data Extraction: Potential biases in selecting studies and extracting data may impact the findings. ***Mitigation:*** *To reduce this risk, we adhered strictly to our peer-reviewed protocol. Data extraction was carefully documented and reviewed iteratively by multiple authors, ensuring consistency. Although the majority of*

data extraction was performed by a single author, this was cross-verified by another author and an external reviewer to enhance accuracy.

- **External Validity**

  Generalizability: The exclusion of non-English papers and those published before 2000 may limit the generalizability of our findings. ***Mitigation:*** *These limitations are acknowledged as common in systematic reviews, and we have focused on a comprehensive analysis of available English-language studies to provide valuable insights. Future research should consider expanding the scope to include non-English literature to enhance generalizability.*

2.5 Conclusion

This chapter provided an in-depth review of the state of research on software modeling tools, focusing on the user experience and socio-organizational aspects through two systematic literature review conducted in 2020 and 2024. By synthesizing findings from 41 and 17 selected papers respectively, the SLRs highlighted the existing challenges, evaluation methods, and gaps in the current literature.

Key takeaways include the need for a more standardized taxonomy of UX challenges in modeling tools, the importance of triangulating evaluation methods for more robust findings, and the necessity of considering non-technical aspects, such as organizational and social barriers, in future research. The review also emphasized the importance of addressing emotional and marketing issues in UX, areas that are often overlooked but critical for broader adoption.

This chapter sets the foundation for the subsequent empirical studies in this thesis, by identifying research gaps and providing a structured understanding of the landscape, which will be further explored and addressed in the later chapters.

Chapter 3 Research Method

In this chapter we explain our research design and how we address our research questions, our selected methods, and the rationale behind choosing them.

Every research approach is based on their fundamental sets of assumptions about the world and knowledge. In adopting a pragmatic philosophy, we view the world as a phenomenon with social order (ontology). Our chosen approach emphasizes a pragmatic stance that blends both objective and subjective elements, allowing for a more comprehensive understanding of this phenomenon through a flexible and adaptive interpretation (epistemology) (Bhattacherjee, 2012; Saunders et al., 2019). We opted for a pragmatism approach because it aligns with our research goals and the complex nature of our study, enabling us to employ a flexible combination of quantitative and qualitative methods. This approach allows us to address the practical implications of our research, emphasizing problem-solving and providing insights that can be applied in real-world contexts. Our interest lies in a comprehensive exploration of user experience – a predominantly subjective concept. Additionally, we aim to develop an adoption framework with practical applications, addressing the challenge of limited adoption of software modeling practices by developers.

Our study employs a hybrid research approach. The majority of data gathering takes empirical approaches (interview, survey), complemented by theoretical approaches to analyze and synthesize the results.

We will describe our philosophy and choices mostly based on the research onion as it is a common approach taking by many researchers. The **Research Onion** is a framework developed by Saunders et al. (Saunders et al., 2019) to help researchers systematically plan and structure their research methodology. It visually represents the layers of decision-making involved in the research process, allowing researchers to understand the various components that influence their approach. The Research Onion consists of several layers, each addressing different aspects of the research design. By systematically considering each layer, researchers can ensure a coherent and logical alignment between their research philosophy, approach, strategy, and methods, leading to more robust and reliable research outcomes.

As our research is situated in an interdisciplinary domain, we also benefit from definitions and guideline for empirical research design in software engineering (Wohlin & Aurum, 2015). Figure 18 demonstrates different layers and choices that we have in performing the research.

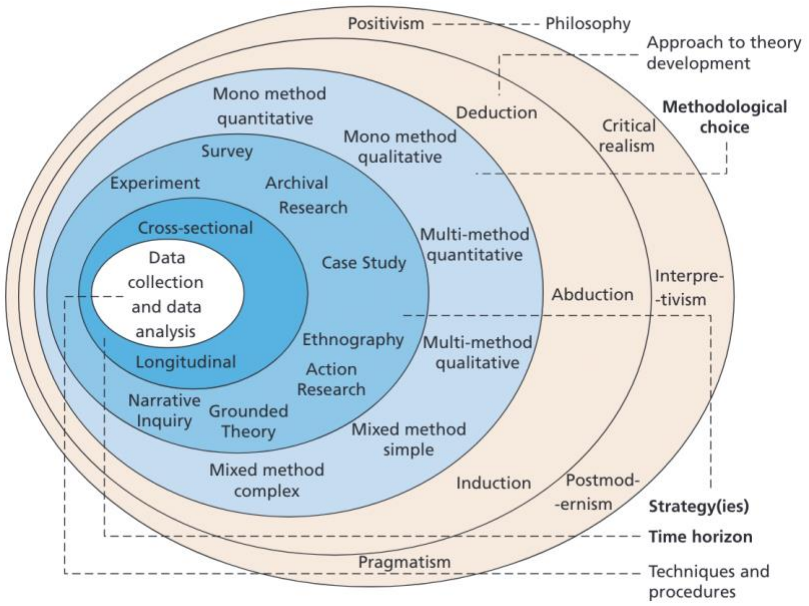


Figure 18: The research onion (Saunders et al., 2019)

Our research is following multiple purposes in its design. In the early stages of the research, when leveraging literature review we are interested in "gaining insight about the topic (Saunders et al., 2019)", our research falls in the exploratory research category. In later stages of the study such as shaping our taxonomy for adoption barriers, it will be more descriptive as it includes, "an accurate profile of a concept (Saunders et al., 2019)" and "careful observation and documentation (Bhattacherjee, 2012)".

As argued by many researchers (Méndez Fernández & Passoth, 2019; Wohlin & Aurum, 2015), today's complex environment, requires more than one research process to uncover all aspects of the issue. Therefore our research will benefit from mixed-method design, to enable us to investigate the topic in more than one aspect (Headley & Plano Clark, 2020), to provide a greater depth and breadth of information (Almalki, 2016) and also to compare results of each

method to see if they support each other (Saunders et al., 2019). Triangulation design provides a richer data in comparison to a mono-method design (Saunders et al., 2019).

3.1 Research Strategy

In this study, we followed an exploratory mixed-method research process to gain a better understanding of developers' experience with software modeling tools and to answer the research questions stated in 0. Using a mixed-method methodology enables a researcher to cover sufficient depth and breadth in their research. This approach not only facilitates the generalization of findings but also allows for broader implications, extending the relevance of the researched issues to the entire population (Dawadi et al., 2021). There are different types of mixed-methods design: 1. Convergent Parallel Mixed-Methods Design, 2. Explanatory Sequential Design, 3. Exploratory Sequential Design (Dawadi et al., 2021). In our study, we opted to use "Exploratory Sequential Design", which follows a three-phase research structure guided by the constructivist principle. In the initial phase, the researcher digs deeply into an issue, transitioning to the post-positivist principle in the subsequent phase to identify, measure variables, and analyze statistical trends (Creswell & Plano Clark, 2018).

We will describe our research strategies and methods in details for each stage of the study in the following sub-sections.

3.2 Literature Review Process

As the first step of our research, we used *archival research* to build our knowledge upon other researchers' work. This part was performed by conducting two systematic literature reviews that are extensively presented in Chapter 2.

3.3 Case-Study: An initial survey on flow experience and our UmpleOnline tool

After conducting a first literature review, it became apparent that emotional factors have been relatively overlooked in the existing research landscape. In an effort to address this gap and enhance our understanding of MX, we conducted a survey to measure flow experience and intrinsic motivation of a software modeling tool (UmpleOnline). This facet of our research aligns with the principles of case study research, characterized by a thorough exploration of a concept or phenomenon within its real-world context (Saunders et al., 2019). In software engineering, the case study is evaluation of a software engineering phenomenon like a tool, methods, project, product, technology in its real-life setting (B. A. Kitchenham, 1996; Runeson et al., 2012).

In our study, we focused on a software modeling tool, namely UmpleOnline for its convenience. Our methodology closely mirrors the approach taken by Kuusinen et al. (Kuusinen, 2016) , who investigated the flow experience and intrinsic motivation in a GUI designer tool to increase the understanding of developer experience. We adopted the Short Dispositional Flow State Scale (SDFS-2) (Jackson et al., 2008) questionnaire for assessing flow experience. The SDFS-2 measures nine dimensions of flow, namely challenge-skill balance, action awareness, clear goals, unambiguous feedback, concentration on task, sense of control, loss of self-consciousness, transformation of time, and autotelic experience. We also utilized some IMI questionnaire items (Kuusinen, 2015) to assess the intrinsic motivation of users. For assessing user perception of the product, we asked two open-ended questions, based on the survey questions in the literature (Ryan, 1982). The consent form and survey questions can be found in Appendix 2: Flow Survey Questions.

Before starting this research project, we sought approval from the University's Research Ethics Board (REB). The certificate of approval is available in Appendix 1: Flow Survey Ethical Certificate.

We followed *Systematic Random Sampling (Baltes & Ralph, 2021)* as our sampling method. Our target population was UmpleOnline users, and we were interested in their sense of flow experience while using UmpleOnline. To achieve this, we instrumented UmpleOnline to

display a study invitation message to every fourth user who had spent more than 5 minutes on the platform and completed at least 10 edits on a model. This approach was chosen to ensure that we approached users who had engaged sufficiently with the tool, making it pertinent to inquire about their flow experience. Participants did not receive any compensation for their time in this study. the decision not to offer compensation was based on the nature of the research context and the goal of encouraging genuine, voluntary participation.

3.3.1 Data Analysis Process for the Flow Survey

We imported the data into an Excel spreadsheet, where we numerically recorded responses to all questions. For instance, Likert Scale responses were assigned values from 1 (Strongly Disagree) to 5 (Strongly Agree). In the spreadsheet, we calculated basic statistics (count, mean, standard deviation, and other metrics) for each question, as well as correlation analysis of items in the survey. Additionally, we created a range of charts and tables, which are presented in Chapter 4.

3.4 Grounded Theory Exploration

Grounded theory, a qualitative research approach, offers variants tailored to different research contexts and philosophical orientations. The choice among these variants hinges on the researcher's specific goals and philosophical stance. The primary variants include: Classical Grounded Theory (Strauss & Corbin, 1990), Straussian Grounded Theory (Corbin & Strauss, 2008), and Constructivist Grounded Theory.

We chose to employ Constructivist Grounded Theory (C-GT) due to its practical and interpretive orientation (Charmaz, 2006), which aligns seamlessly with the pragmatic nature of our research. This variant offers a valuable and practical guide that resonates with our aim to interpret and understand the subjective meanings attached to phenomena in a flexible manner. The emphasis on the researcher's role in shaping the research process and outcomes within C-GT suits our pragmatic approach, allowing us to navigate the complexities of our research questions while acknowledging the influence of our perspectives on the investigation.

We opted to use interviews as our data gathering methodology. Grounded theory methodologies are especially suited for the application of intensive qualitative interviewing. “Both grounded theory methods and intensive interviewing are open-ended yet directed, shaped yet emergent, and paced yet unrestricted (Charmaz, 2006)”. Employing interviews as a data collection method has proved instrumental in studying social aspects of software and understanding peoples’ opinions, thoughts and feelings (Hove & Anda, 2005; Sharp et al., 2016).

3.4.1 Participant Recruitment Process for the Grounded Theory Study

Before starting this research project, we sought approval from the University's Research Ethics Board (REB). The certificate of approval is available in Appendix 3: Interview Ethical Certificate.

Our participant pool comprised developers, selected through purposive sampling to ensure a targeted and insightful representation (Baltes & Ralph, 2021). Our eligibility criteria specified a minimum of 5 years of industrial development experience for participants.

We offered interview invitations to all developers whose eligibility criteria was confirmed through an online screening survey. This survey included questions about their years of experience, work domain, and preferred interview times. This process served the dual purpose of confirming participants’ eligibility and allowing us to prepare for the interview. The questionnaire, crafted using SurveyMonkey, can be found in the Appendix 4: Interview Protocol.

The interview invitation was advertised through the social networks and personal contacts of the researcher and her supervisor.

We aimed for a balanced sample of participants including modelers and non-modelers, in order to achieve a diverse range of developer perspectives. Following initial interviews with random developers who turned out to be non-modelers, we reached out to commercial software modeling tool vendors such as Astah and Magic Draw to assist us in identifying potential participants who were active users of modeling tools. The recruiting and interviewing process was done concurrently over an 8-month period, spanning from September 2022 to April 2023. This iterative process persisted until we achieved data saturation: As the interviews progressed, we analyzed the data in parallel with data collection. After each set of interviews, we reviewed

the findings to identify emerging themes and patterns. Once we noticed that new interviews were only reinforcing existing themes without introducing new concepts, it indicated that saturation was likely being reached.

3.4.2 Interview Process

We conducted remote interviews to enhance flexibility and facilitate engagement with developers globally, ensuring a diverse and inclusive representation. The choice of a semi-structured interview format, incorporating both open-ended and specific questions, aims to not only unveil anticipated information but also uncovering unexpected insights (Seaman, 1999). The interview protocol, which evolved during our data collection, comprised questions about participants' experiences, and opinions related to software modeling, the challenges they faced, and their perceptions of the value of modeling in their development processes. The interview protocol is in Appendix 4: Interview Protocol.

The interviews were conducted using the Zoom platform. All interviews were hosted by both researcher and her supervisor. Each interview session lasted approximately 30 minutes.

The process was approved by the University of Ottawa Research Ethics Board. Prior to each interview, we sent an informed consent form to each interviewee. The first step in each interview was to ask orally whether each interviewee provides consent to proceed.

All interviews were video recorded, but the videos were used only for transcription purposes.

3.4.3 Data Analysis Process

Each interview was transcribed using tools including Microsoft Word's "Dictate" feature and Otter.ai. The transcript was then coded.

The process of data analysis for this study was facilitated by the use of Dedoose (*Dedoose*, n.d.), a software application designed for analyzing qualitative research data. Dedoose was instrumental in managing and organizing the interview transcripts, enabling efficient coding, and assisting in the identification and visualization of emerging themes. We also used an Excel spreadsheet to gain a tabular perspective on the data.

To ensure the reliability of the analysis, both researcher and supervisor independently coded the data and then compared their results, discussing and resolving any discrepancies through consensus.

Regarding the coding process, we followed the grounded theory coding approach prescribed by Charmaz (Charmaz, 2006), which includes two phases: firstly, an initial phase wherein each word, line, or data segment is assigned a name, and secondly, a focused and selective phase that employs the most significant or frequently used initial codes to categorize, synthesize, integrate, and structure data. In the initial coding phase, the aim is to stay receptive to all potential theoretical directions suggested by the data. Subsequently, focused coding is employed to identify and refine the most significant categories within extensive data sets. Finally, we performed *theoretical coding* to conceptualize relationships between categories that we had developed in our focused coding.

Figure 19 shows some of the codes and sub-codes as shown by the Dedoose software.

We present the results of our grounded theory research in Chapter 5.

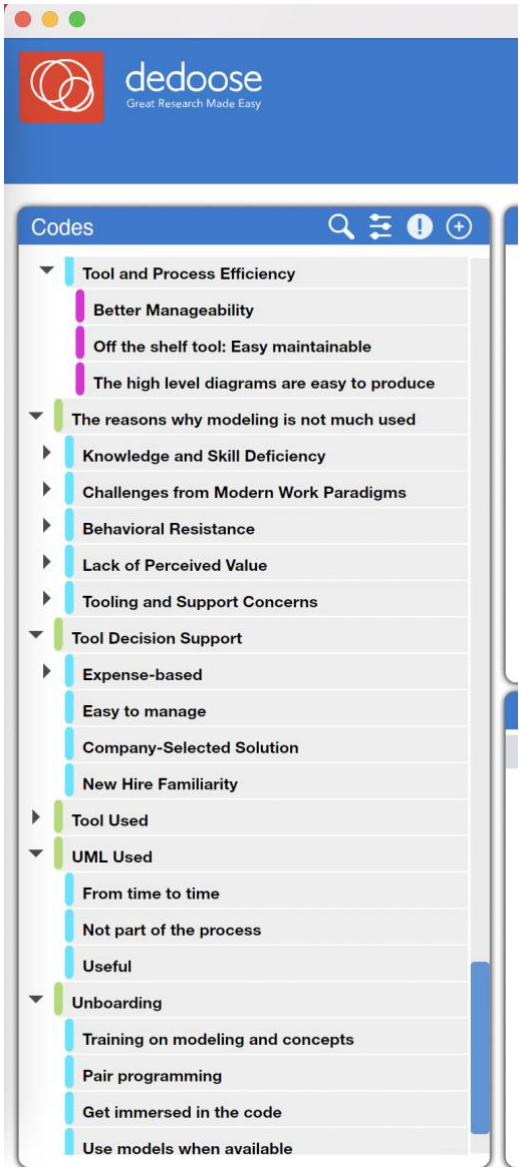


Figure 19: A portion of codes and sub-codes from our research, as presented in Dedoose

3.5 Survey-Based Validation

In the concluding phase of our research, we crafted a survey incorporating the themes identified in the previous stages of the study. This strategic approach allowed us to gather data from a more extensive pool of developers, facilitating the generalization and confirmation of our

findings. The survey encompassed questions regarding both perceived benefits and challenges faced by developers to mitigate potential bias.

The survey was developed using SurveyMonkey. We conducted a pilot study to refine the questions, and it was then disseminated through our social media channels. We asked the participants about the level of experience they have as a software developer, the extent of modeling they perform, their geographic region, their gender, the modeling tools they use, and the extent to which they agree with statements describing 12 benefits and 32 challenges of modeling. These 44 statements were derived from our grounded theory analysis, and agreement was expressed using a 5-point Likert Scale from Strongly Agree to Strongly Disagree, with Neutral in the middle.

The complete survey is shown in Appendix 6: Final Survey Questions.

3.5.1 Participant Recruitment Process

Before starting this survey, we sought approval from the University's Research Ethics Board (REB). The certificate of approval is available in Appendix 5: Final Survey Ethical Certificate.

Our participant pool comprised developers, selected through purposive sampling, with the same eligibility criteria as our interview process: a minimum of 5 years of industrial development experience for participants.

We extended invitations through our social networks, personal channels, and the contacts of both the researcher and supervisor, aiming for inclusivity and diversity among participants. We also reached out to our interview participants and invited them both to fill out the survey and to help us distribute the survey to additional developers. Data collection occurred between December 2023 and January 2024.

3.5.2 Sample Size Calculation

To ensure that our study results are statistically significant and representative of the population, we calculated the required sample size using a confidence level of 90% and a margin of error of 10% and the result was 67.65.

Therefore, the sample size required is approximately 68 participants. We included 86 participants in our study despite needing only 70 to account for potential dropouts and non-compliance, thus ensuring that our results would remain robust and representative of the target population.

3.5.3 Data Analysis Process

We downloaded the data into an Excel spreadsheet where the responses to all questions were recorded numerically. For example, the Likert Scale responses were given values from 1 for Strongly Disagree to 5 for Strongly Agree, On the spreadsheet we computed basic statistics (n, average, standard deviation, and various other values) for each question. We also enabled Excel's filtering mechanism to allow us to select subsets of the participants.

From the Excel data we generated a variety of charts and tables that we present in Chapter 6. We also exported the Data to R in order to perform various statistical analyses that are reported in that chapter.

We chose to employ parametric tests for our data analysis, despite the ongoing debate surrounding their application to ordinal Likert scale values. Literature indicates that parametric testing is both powerful and robust, even when traditional assumptions are not fully met. As Norman (G. Norman, 2010) asserts, "Parametric statistics can be used with Likert data, with small sample sizes, with unequal variances, and with non-normal distributions, without concern for reaching wrong conclusions." This evidence supports our decision to utilize parametric methods in our study.

3.6 Conclusion

In this chapter, we described the details of how we conducted our study on developers' experiences with software modeling tools.

Our research serves multiple purposes, evolving from gaining insights to creating an accurate profile of our concept. Recognizing the complexity of our research environment, we

used a mixed-method design, specifically the exploratory sequential design. This design unfolded in three phases, allowing us to deeply explore the issue and subsequently identify variables and analyze statistical trends.

Our research strategy follows an exploratory mixed-methods process, offering a comprehensive understanding of developers' experiences.

We started with an initial systematic literature review (SLR) and analyzed by employing a mix of quantitative and qualitative methods. This was complemented by a second SLR when we later encountered new topics that needed investigating. Moving to the case study phase, we focused on emotional factors overlooked in existing research. The survey measured flow experience and intrinsic motivation of developers using UmpleOnline.

Our grounded theory exploration embraced Constructivist Grounded Theory (C-GT), aligning with the pragmatic nature of our research. Interviews were our chosen data gathering method, suited for intensive qualitative interviewing.

The final phase involved a survey with themes identified in prior stages, designed to confirm and generalize our findings.

In summary, our research methodology is a carefully crafted, adaptable framework designed to uncover the nuances of developers' experiences with software modeling tools. The credibility of this research is strengthened by using *triangulation of methods and data sources,* which is a common approach for presenting rigorous qualitative research (Creswell & Poth, 2016; Flick et al., 2004; Henry, 2015; Runeson & Höst, 2008). Each phase is systematic and transparent, minimizing bias and ensuring the reliability of our findings. This methodological journey sets the stage for subsequent chapters, where we unfold insights and implications derived from our multifaceted approach.

Table 7 summarizes our chosen research methods, data collection and data analysis methods, to answer each research questions.

Table 7: Research method summary

Research Question	Data Collection Method	Data Analysis Method
RQ1, RQ2, RQ3	Literature Review	Statistics/ Content analysis (Thematic analysis)
RQ4, RQ5	Questionnaire	Statistics/ Content analysis (Thematic analysis)
RQ6, RQ7	Semi-structured Interview	Grounded Theory
RQ6, RQ7	Questionnaire	Statistics/Content analysis
RQ8	The overall findings	

Chapter 4 Results of Case-Study: An initial survey on flow experience and UmpleOnline tool

The content in this chapter is an extended version of the paper presented in the HuFaMo (Human Factors in Modeling) workshop of the MODELS 2022 conference and published in the proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems (Kalantari & Lethbridge, 2022b). In particular, we have updated it to include many survey responses obtained since we submitted the paper to that workshop.

Following our systematic literature review, the results of which were discussed in Chapter 2, we observed that the papers we analyzed uncovered usability issues, utility issues (needed functionality not available), and other categories of MX issues such as lack of reliability and excess cost. However, an important category of MX issues was not analysed in the existing literature: emotional factors. Many researchers addressed the importance of emotional factors in user experience by taking different approaches such as affective design (Hassenzahl, 2008), Kansei engineering (Nagamachi, 2010), emotional design (D. Norman, 2004), and emotional human-technology interaction (Saariluoma & Jokinen, 2014), all highlighting users as emotional-experiencing beings. Therefore, deeply understanding emotional experiences soothes the path to improving overall UX.

One aspect of emotional experience is flow. Various authors have pointed out that a higher degree of flow experience is associated with a more positive emotional state in users (Peifer et al., 2022) . Additionally, a higher positive emotional state in software engineers is directly correlated with increased productivity (Wróbel, 2013) .

Although we assume flow experience as an indicator of emotional state, some studies correlate flow experience with other qualities of user experience, such as perceived ease of use and usefulness (Agarwal & Karahanna, 2000), hedonic and utility characteristics (M.-H. Huang, 2003), flexibility and modifiability (Webster et al., 1993), and general satisfaction (Rau et al., 2006; Woszczynski et al., 2002). The study reported in the current chapter aims to measure flow experienced by users of a software modeling tool – in other words, how well the modeling tool enables users to feel they are focused while doing productive, enjoyable work.

This study focuses on the flow experience of users of UmpleOnline.

4.1 What Is Flow Experience

Flow experience, which was first characterized by Csikszentmihalyi (Csikszentmihalyi, 2000), is the process by which individuals engage in autotelic activities, activities that are so engrossing and rewarding that they are usually done for their own sake. Csikszentmihalyi defined certain dimensions of flow: (1) a balance between the challenge of the task and skills of the individual, (2) a merging of action and awareness, (3) clear perceived goals, (4) unambiguous feedback, (5) focusing on the task at hand, (6) a sense of control of the activity, (7) a loss of self-consciousness or a reduced awareness of self, (8) time transformation, and (9) an autotelic, intrinsically rewarding experience. Since introducing this concept, the dimensions have been operationalized and measured variously in different domains. Different sets of dimensions are applied by many studies to assess flow experience.

Due to the clear positive effects of flow experience like increasing productivity, learning and positive experience, flow research is growing in many domains (Zhang & Finneran, 2005). Flow experience has been assessed in areas including physical activity, education, art, gaming, and computing domains. In the latter it mostly has been studied in human-computer interaction and online-environment studies (Hamari & Koivisto, 2014). HCI (Human-Computer Interaction) researchers use flow experience to understand human behavior in using computers and thus inform design, training and user experience (Zhang & Finneran, 2005).

Kuusinen (Kuusinen, 2016) examined flow experience in the context of developer experience using development tools: Qt Creator (An IDE) and Vaadin Designer (a graphical user interface designer tool), and their results suggest that flow experience is a significant predictor of developer experience in general. It seems therefore logical to study flow in modeling tools to improve modeling experience. We also took the opportunity to ask a few other questions about the user experience of UmpleOnline itself.

4.2 Results of Flow Experience Survey

The research method for this study, a survey, is described in Section 3.3, along with all the other research methods presented in this thesis.

The survey was available on UmpleOnline from November 2021 until July 2024, during which time we received 63 valid responses out of 66 responses. Requests to do the survey appeared occasionally after users had used UmpleOnline for a while (as described in Section 3.3). We screened for invalid responses by including questions where random answers would be inconsistent, such as claiming to be both a grade-school student and a graduate student. We also screened for obvious patterns of responses by participants (for example by giving the same answer to all questions). Additionally, we checked for missing values and removed 2 more responses from two participants who had five and seven missing responses in their Likert scale questions. For three other questions with one or two missing values, we allowed slight variations in the sample size when calculating statistics.

4.2.1 Descriptive Analysis of Respondents' Profiles

The chart in Figure 20 indicates that the majority of respondents are undergraduate students, making up the largest group with 45% participants. Practitioners in the industry form the second-largest group with 22% respondents, followed by graduate students with 12% responses. University educators and individuals in the "Other" category also contributed to the survey, with 11% and 9% respectively, while grade school students had the fewest responses (3%).

The "Other" category includes people who indicated they were a 'Professional NEET' (i.e. not in education, employment or training), a 'former practitioner', two 'former students', an 'independent researcher' and a person who described themselves as being an officer and mentor of a non-profit organization.

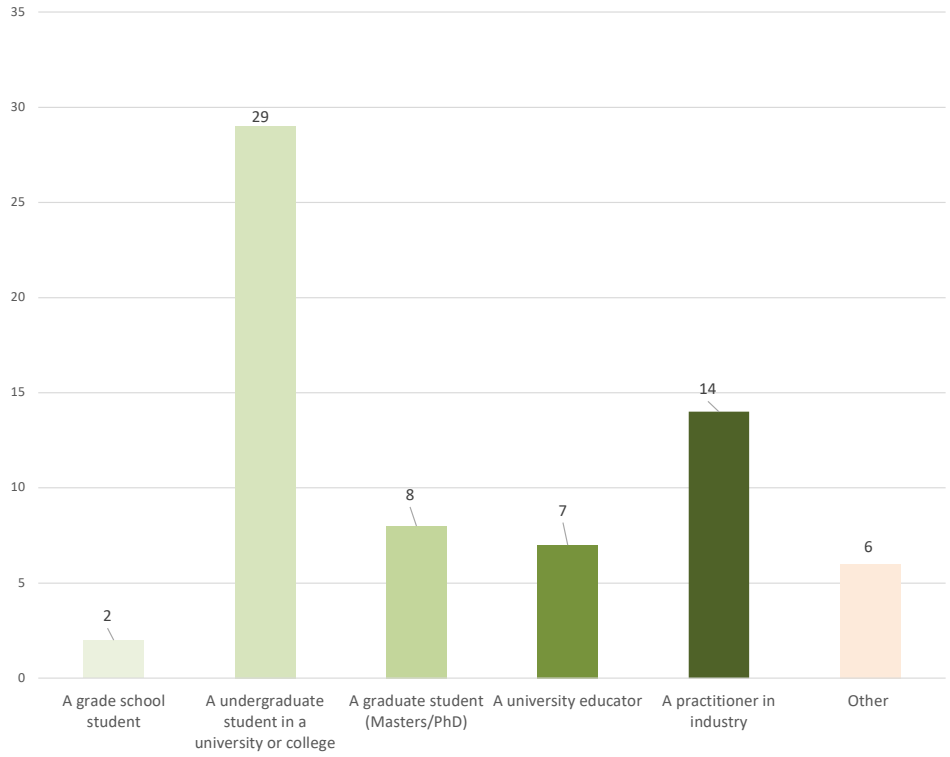


Figure 20: Survey Respondent distribution by current status

We also asked participants about their experience level which we defined as follows:

- Inexperienced (I have only created a few very simple models or have created models only in academic courses)
- Medium
- Expert (I have been using modeling tools extensively in professional practice for at least 6 months)

The pie chart in Figure 21 illustrates the distribution of survey respondents based on their level of experience. The largest group, representing 52% of respondents, is categorized as inexperienced, with a total of 33 participants. The medium experience group makes up 32% of the respondents, totaling 20 participants. The smallest group, comprising experts, accounts for 16% of the respondents, with 10 participants. This distribution highlights that the majority of the survey participants are either inexperienced or have medium-level experience, with a smaller proportion being experts.

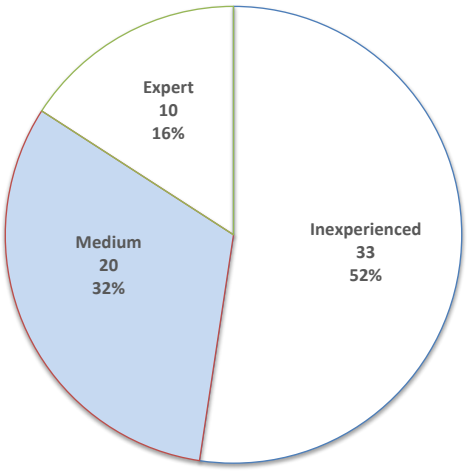


Figure 21: Survey Respondent distribution by their experience level in modeling

The bar chart in Figure 22 illustrates the geographical distribution of survey respondents across various regions. We mapped each respondent's country to its region. North America leads with the highest percentage of respondents at approximately 48%, followed by Europe with 32%. South America accounts for 13% of the respondents, while Oceania and the Middle East and Africa each represent 3%. East Asia has the fewest respondents, making up only 2%. This distribution highlights the geographic spread of the survey's reach.

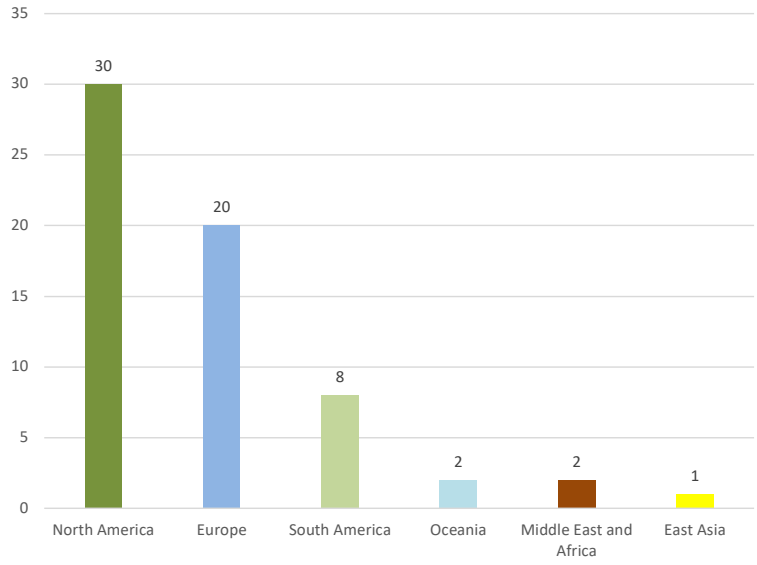


Figure 22: Survey Respondent distribution by geographic distribution

The pie chart in Figure 23 illustrates the distribution of modeling tools across three capability categories. The categorization primarily emerged from our grounded theory analysis; more detailed explanations can be found in Section 5.9.

The largest segment, comprising 77% of the tools, falls under the "Analysis and Generation Capable Tools" category. These tools are designed to provide advanced analytical and generation capabilities, making them highly functional for complex modeling tasks. The second-largest segment, representing 21% of the tools, is categorized as "Diagram Focused Tools." These tools primarily focus on diagram creation and visualization, offering specialized features for visual modeling. The smallest segment, making up only 2% of the tools, consists of "Informal Tools," which are likely used for more casual or less-formal modeling purposes.

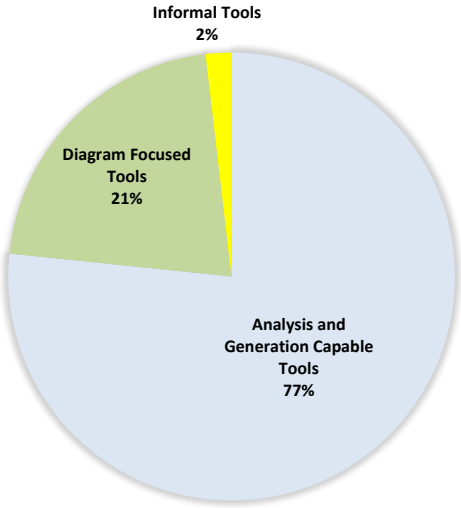


Figure 23: Survey Respondent distribution by used tool category

Figure 23 provides a visual summary of these categories and their respective proportions among survey respondents. Meanwhile, Table 8 offers detailed information on the specific tools classified under each category. This table highlights the distinction between informal tools, diagram-focused tools, and those with full analytical and generation capabilities.

Table 8: Details on tool categorization in flow survey

Informal Tools	**Diagram-Focused Tools**	**Analysis and Generation Capable Tools**	**Maximally Formal Tools**
Hand-written	Argo UML	Papyrus	USE
Confluence	PlantUML	Umple	
	Diagram.net	MagicDraw	
	Lucidchart	Visual Paradigm	
	Visio	Enterprise Architect	
	Smart Draw	Astah	
	Umbrello	Eclipse based modeling tools	
	draw.io	Rational Rose	
	UMLET	Capella	
		Rhapsody	
		RSA	
		Visual Studio	

In the survey, we asked participants why they chose Umple over other modeling tools. 35% of respondents indicated that they chose Umple because it was required by their

professor/supervisor/teammate, 24% indicated that it was the only tool that they were familiar with. 41% mentioned that they found Umple through a link; 24% reported that they needed one or more particular features that seem only available or are best in Umple. And Umple is introduced as participants' preferred tool by 14 users (22%). We also had a free-text option to allow our participants to share any other ideas about this question, which was filled by 8 participants. Respondents provided various reasons for selecting Umple, reflecting a range of motivations and experiences. One respondent mentioned enjoying "marathon programming languages," while others highlighted practical benefits, noting that "typing is faster than drawing," and describing it as "handy" and "well established." Some are exploring Umple for educational purposes, such as preparing for a class, while others see a broader potential, stating they "see a wide future for it" and find it "ample and versatile."

The ease of use and accessibility of Umple are significant factors, with users appreciating that it is "very easy to use and show to my students," as well as being "free and simple to learn." The open-source nature of Umple also appeals to users. Some have championed its use professionally, implementing it in production systems at Fortune 500 companies.

These responses highlight the diverse appeal of Umple, from practical benefits and educational uses to its future potential and open-source advantages.

We asked participants which Umple tools they have used. As expected, all participants selected UmpleOnline as one of the tools, since the survey was initiated from a button in UmpleOnline, but 16% also selected IDE Umple Plugins for Eclipse or VSCode, and 8% indicated they had used the Umple command-line compiler.

We also asked participants what is the main task that they use Umple for. The answers were formed as four pre-defined options, with an "Other" option to enable respondents to answer freely. The bar chart in Figure 24 illustrates the distribution of responses. The most common use, reported by 40 respondents (63%), is "Drawing diagrams," highlighting Umple's strong capability in visual modeling. 14 respondents (22%) used it for generating a small functioning system, 6 people (10%) used it to build a large complex system, and only 2% for analyzing a model for errors and weaknesses. There may be several reasons for most people only using the

tool for diagramming: it could be due to features like analysis or code generation being less known or not required for the user's goals.

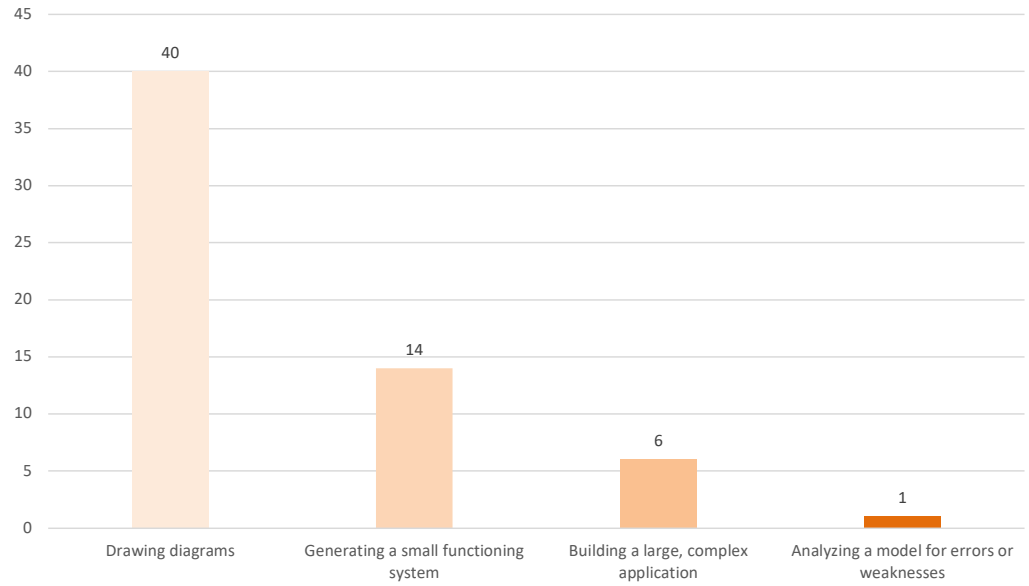


Figure 24: Primary Uses of Umple by Respondents

4.2.2 Results: Descriptive Analyses of Flow Questions

As discussed in Section 3.3, to capture users flow experience, we employed the SDFS-2 and IMI questions. Using these, we measured each construct of flow with one measurement. One of the most-cited factors contributing to flow experience is the balance of challenge perceived by users and the skill that they have in the task (Cohen & Bodner, 2019; Marin & Bhattacharya, 2013; Rheinberg & Engeser, 2018; Zhang & Finneran, 2005). The high challenge inherent in software engineering has been documented as contributing to developer motivation (France & Rumpe, 2007).

We calculated mean, standard deviation, the percentage of respondents who gave the two top-most Likert scale responses (agree and strongly agree, also scored 4 and 5 in our other calculations), as well the percentage who responded with the bottom-two responses (disagree and strongly disagree, scored 2 and 1). We summarize results for each subscale in Table 9.

All mean scores were above the neutral point (>3.0), with a mean of 3.38 which could be interpreted as a feeling of a moderate state of flow in respondents. The flow dimension with the most respondents saying they agree or strongly agree, is *Loss of self-consciousness* (71%), followed by *Clear goals* (68.3%). On the other hand, *Transformation of time* (29.5%) and *Action awareness* (27.4%) are the flow dimensions with highest percentages of respondents choosing to disagree and strongly disagree, although in both cases the agreement still exceeded the disagreement.

Table 9: Results of SDFS-2 subscales, measuring agreement on a scale of 1-5 with various flow statements

Flow dimensions	SDFS-2 item	Mean	St. Dev.	% Top2	% Bottom 2
Q8 Challenge-skill	I feel I am competent enough to meet the high demands of the situation	3.32	1.12	52.4	23.8
Q9 Action awareness	I do things spontaneously and automatically without having to think	3.06	0.97	37.1	27.4
Q10 Clear goals	I have a strong sense of what I want to do	3.70	1.01	68.3	14.3
Q11 Unambiguous feedback	I have a good idea while I am performing about how well I am doing	3.56	0.98	61.9	15.9
Q12 Concentration on task	I am completely focused on the task at hand	3.71	0.85	66.7	7.9
Q13 Sense of control	I have a feeling of total control	3.35	1.03	52.4	22.2
Q14 Loss of self-conscious-ness	I am not worried about what others may be thinking of me	3.81	1.16	71.0	14.5
Q15 Transformation of time	The way time passes seems to be different from normal	3.02	1.23	36.1	29.5
Q16 Autotelic experience	The experience is extremely rewarding	3.15	0.99	35.5	19.4

We also utilized some items from the IMI questionnaire to measure intrinsic motivation as an effective factor of flow experience. Intrinsic motivation is often contrasted with extrinsic

motivation in which the incentive resides in the expected external results rather than the activity itself (Rheinberg & Engeser, 2018). We selected the following IMI subscales, similar to (Kuusinen, 2016): interest/enjoyment, perceived competence, perceived choice, and effort/importance. The results are summarized in Table 10.

There is a moderate disagreement (with an average of 2.67) regarding the "I have no choice" statement (which is deliberately expressed in an inverse sense), indicating that participants felt they do indeed have choice; this is promising, since tool choice might naturally be sometimes limited by workplace or academic demands (i.e. the team is using the tool, or the professor assigns it.)

Table 10: Results of IMI subscales, measuring agreement on a scale of 1-5 with aspects of motivation

IM dimensions	IMI item	Mean	St. Dev	% Top 2	% Bottom 2
Q17 Interest / enjoy-ment	I enjoy using Umple very much.	3.56	1.01	54.0	11.1
Q18 Perceived competence	I am pretty skilled in using Umple.	2.76	0.87	14.3	38.1
Q19 Perceived choice	I use Umple because I have no choice. [inversely phrased]	2.67	0.95	17.5	44.4
Q20 Effort / importance	It is important to me to do well while using Umple.	3.63	0.89	58.7	6.3

4.2.3 Results: Correlation analysis of flow survey responses

This section explores the relationships between various dimensions of flow and intrinsic motivation as captured in the survey. By examining how different aspects of user experience, such as experience level, task type, and emotional engagement, correlate with flow states, we can gain insights into the factors that facilitate or hinder effective and enjoyable use of modeling tools like Umple. Figure 25 shows the correlation values of the items, with questions presented in a shortened format to improve readability. Each cell in the heatmap represents the Pearson correlation coefficient between two variables, with green shades indicating positive correlations and red shades indicating negative correlations. The intensity of the color reflects the strength of the correlation, ranging from -1 (strong negative correlation) to +1 (strong positive correlation).

4.2.3.1 Key Observations of correlations between all items:

The following are the strong positive correlations:

- **Q16 Autotelic Experience and Q17 Enjoyment (0.80)**: This strong positive correlation suggests that respondents who find their tasks intrinsically motivating (autotelic) also tend to report higher levels of enjoyment. This relationship is crucial, as it implies that fostering an autotelic experience can directly enhance user satisfaction and engagement.
- **Q10 Clear Goals and Q12 Concentration on Task (0.59)**: The positive correlation between clear goals and concentration indicates that when respondents have clear goals, they are more likely to be able to concentrate on their tasks. This highlights the importance of clarity in task objectives for maintaining focus.
- **Q12 Concentration on Task and Q13 Sense of Control (0.52)**: Respondents who are more focused on their tasks tend to feel a greater sense of control. This relationship is consistent with the concept of flow, where concentration and control are key components.

The following are the moderate positive correlations:

- **Q15 Transformation of Time and Q16 Autotelic Experience (0.45)**: The feeling of time passing differently (a transformation of time) is moderately correlated with autotelic experience, suggesting that immersion in the task may alter time perception.

The following are some notable negative correlations:

- **Q16 Autotelic Experience and Q19 Perceived Choice (-0.42)**: This negative correlation implies that as the autotelic experience increases, the lack of perceived choice decreases. It could suggest that when tasks are intrinsically motivating,

sense of control and having a choice is higher, potentially because the task is enjoyable.

4.2.3.2 Key observations of correlation between Experience (Q3) and other items

The analysis shows a positive correlation between experience level and all items related to flow and intrinsic motivation, except for the last item: Effort/Importance (-0.03), which exhibits a weak negative correlation. This suggests that while increased experience generally enhances concentration, goal clarity, and perceived competence, more experienced users tend to perceive the effort required as less important, possibly due to familiarity or proficiency, making the effort seem less demanding.

4.2.3.3 Key observations of correlation between Main Umple Task (Q7) and other items

The analysis of the main Umple task shows different correlations with flow dimensions and perceived experiences. The negative correlation with **Q8 Challenge Skill** (-0.12), **Q9 Action Awareness** (-0.11), **Q10 Clear Goals** (-0.10) and **Q11 Unambiguous Feedback** (-0.23) suggests that users involved in tasks like "Drawing diagrams" or "Generating a small functioning system" may experience less clarity regarding their goals and receive less effective feedback. These tasks might not offer enough challenge or clear direction, potentially leading to ambiguity and a reduced sense of accomplishment. This finding underscores the need to refine task definitions and set clearer expectations to enhance user satisfaction and engagement.

Conversely, positive correlations with **Q16 Autotelic Experience** (0.25) and **Q17 Enjoyment** (0.23) imply that more complex tasks, such as "Building a large, complex application" or "Analyzing a model for errors or weaknesses" offer users a more rewarding and enjoyable experience. These tasks likely require higher levels of skill and focus, which align with users' sense of competence, leading to a more satisfying and intrinsically motivating (autotelic) experience.

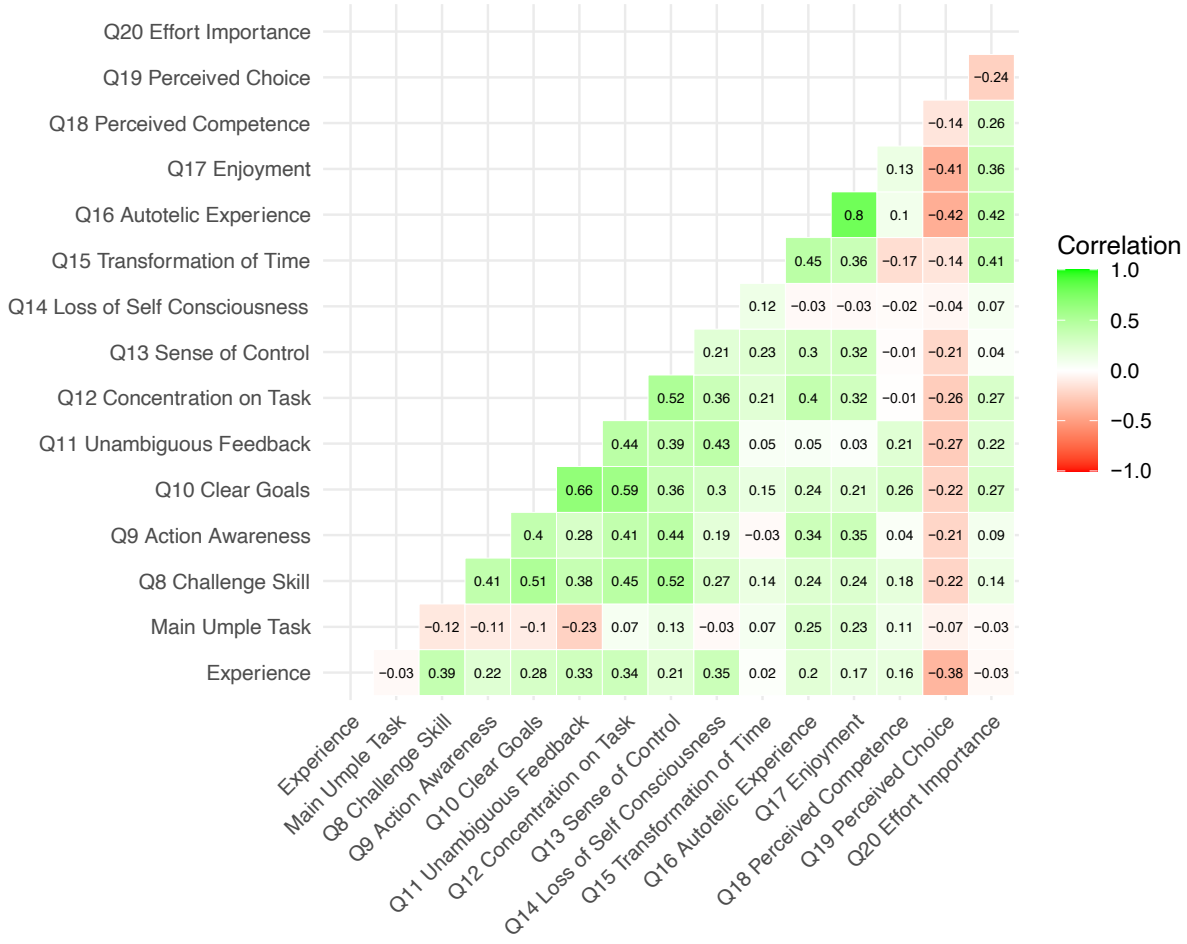


Figure 25: Correlation analysis of flow survey responses

4.2.4 Results: Best Qualities of Umple

There were 36 responses to the free-form question asking about the best qualities of UmpleOnline.

Based on the survey responses, here is a categorized summary of the best qualities of Umple as identified by the respondents, ordered by the number of mentions:

- Code Generation Capabilities (13 mentions):

  Respondents highlighted Umple's robust code generation features, emphasizing its ability to generate code from diagrams and vice versa.

- Ease of Use (9 mentions):

  Umple is praised for its user-friendly interface and simplicity. Respondents mentioned "It is easy to use," and "Simple to use since the very first time".

- Ease of Learning (6 mentions):

  Simplicity and ease of learning are key strengths of Umple. Comments such as "It's easy to learn and it's not polluted with lots of information" and "Easy learning process" were highlighted. One participant stated, "I learned how to use most features in 5-10 minutes by editing examples, compared with thousands of pounds of training courses for competitors."

- Availability of Support, Information and Examples (6 mentions):

  The availability of support, information and examples helps users learn and use Umple effectively. Respondents appreciate the support, the good manual and ready-to-use examples.

- Code/Diagram Duality (5 mentions):

  The duality of code and diagrams is a strong point, allowing users to work with both seamlessly. This is highlighted by this participant's comment: "…To be able to just type what's in your head in an instantly-intuitive language and get a diagram immediately, with minimum graphical edits, is amazing."

- Web-based access (4 mentions):

  Umple's availability as an online tool without the need for installation is appreciated. Comments such as "It has an online version," "does not need any installation" and "Runs inside your browser" emphasize its accessibility.

- Textual Capabilities (4 mentions):

  The ability to use text-based descriptions alongside diagrams is seen as a strength. Respondents mentioned "Text-based description of diagrams" and "Designing UML using text format is so great."

- Diagram Generation (4 mentions):

  Respondents appreciate Umple's capability to generate diagrams. One participant echoed their thought: "…Most of the time I don't even have to touch the mouse. Modeling errors reveal themselves quickly thanks to the instant diagram; I can iterate quick, and test my thoughts"

The other capabilities mentioned by respondents include Umple being free, the ability to execute code within Umple, generating code in multiple languages such as Java and Python, and exporting diagrams to other modeling tools.

4.2.5 Results: Requested Improvements for Umple

Respondents provided valuable feedback on the issues and areas for improvement in Umple, highlighting suggestions for improved usability and utility. There was a total of 29 valid responses to this question.

The main usability concerns included the general suggestion for better UI/UX, faster performance, increased responsiveness, improved error detection, and more intuitive shortcuts. Specific suggestions were made for adjusting diagram layouts, supporting collaborative work, providing more space for diagrams, and providing a cleaner generated code. There were also

calls for more concrete error messages, cleaner code, and better support for uploading and downloading files, including cloud options.

We discussed these comments with the development team. Some of them, such as supporting collaborative work and improved error detection of embedded code are already developed. Others, such as need for faster performance likely result from the end-user's internet connection being slow, since on a fast connection Umple responds rapidly as pointed out by other respondents. Some comments, such as the request for 'cleaner generated code,' highlight areas where Umple is designed to improve upon the shortcomings of other tools. While it's possible that some respondents may not have had the opportunity to compare Umple directly with these alternatives, it is important to note that readable code was a design criterion for Umple to address the issues associated with code generated by other tools. This feedback is valuable and will be considered for future development as we continue to enhance Umple's capabilities.

On the utility side, respondents highlighted the need for generating code in more languages, such as Python and C# and additional diagram types such as network diagram. There were also requests for enhanced round-trip engineering, offering a code-to-Umple option, and improving the diagram rendering engine.

These comments were again relayed to the Umple development team. Some of them, such as the need to generate Python have already been completed. The suggestion for round-trip engineering was not accepted, since one of the stimuli for the development of Umple was the observation that it does not work; this observation has been corroborated by others in the community (Siegl, 2015). The developers made note of the other suggestions.

Respondents expressed the desire for educational features, such as teaching tools, hints, and a comprehensive step-by-step manual.

A few bugs were pointed out too.

Overall, the positive and negative feedback indicates that Umple is a good tool with some room for improvement in both usability and functionality, and more versatility for additional modeling needs.

4.3 Threats to Validity and Limitations

The main threat to the validity of this chapter is internal validity, we tried to mitigate this threat by randomizing participant recruitment from a large pool. We solicit 1 out of every 4 users who meet our criteria for time using UmpleOnline and edits performed in their session. A total of 8457 people were invited to participate, of which only 264 (3.1%) clicked the link, and only 63 submitted valid responses (0.75% of those asked). Since so many websites now ask people to do surveys, we attribute this low response rate simply to survey response fatigue. It is also possible that people who are "in the zone", i.e. with a good sense of flow, are skipping our survey because they want to continue with their task; alternatively, it could be that only the most enthusiastic or most negative-feeling people are more motivated to respond. Any of these factors could be providing a bias.

It does seem that participants were well-distributed among education levels, occupations, tool experience and similar factors therefore reducing demographic bias.

There is a threat to external validity, since the survey only applies to one modeling tool, UmpleOnline. This limitation means that the findings and insights derived from the survey may not be generalizable to other modeling tools or contexts. Consequently, the results might reflect the specific characteristics, strengths, and weaknesses of UmpleOnline rather than providing a comprehensive understanding applicable to a broader range of modeling tools.

To mitigate the threat to construct validity of this research, the majority of survey questions come from validated surveys so the measures and constructs should be considered reliable. We also performed a pilot study to identify any flaws and misunderstandings of our questions. Moreover, we considered "other" options in answers items whenever it was applicable, to include all possible answers.

4.4 Discussion and Conclusion

This chapter presents an extended and updated version of our research initially presented at the Models 2022 conference, with additional data gathered through a survey of UmpleOnline users. We collected and analysed 63 responses. Respondents were from both industry and

academia, with different levels of expertise in modeling and from different countries, so we expect that this set of respondents formed a representative sample of our target group of users.

The survey results revealed a diverse range of user experiences, with a significant portion of respondents identifying as inexperienced with modeling tools. The analysis showed that users generally experience a moderate state of flow while using UmpleOnline, with particular strengths in maintaining concentration and minimizing self-consciousness during tasks. However, areas like action-awareness and time transformation were less pronounced, suggesting opportunities for enhancing the tool's capacity to support deeper flow experiences.

Our findings highlight the importance of intrinsic motivation in shaping user experiences and the **interconnection between Flow and Motivation**. The strong correlations among flow dimensions suggest that enhancing one aspect (e.g., clear goals) can positively impact others (e.g., feedback and concentration). Addressing these interdependencies could improve overall user experience.

The results highlight that simpler tasks may not offer enough challenge or clarity, leading to less-effective feedback and lower perceived competence. Users involved in more complex tasks report higher enjoyment, perceived rewards, and a stronger sense of control, suggesting that task complexity positively influences these aspects of user experience.

The notable negative correlation with feedback quality (Q11) indicates that enhancing feedback mechanisms for simpler tasks could improve users' ability to gauge their performance. Providing clearer, more detailed feedback even for simpler tasks could help bridge this gap.

To enhance user satisfaction and engagement, it is essential to ensure that tasks, regardless of their complexity, have well-defined objectives and robust feedback systems. This approach will help users feel more competent, focused, and in control, ultimately leading to higher levels of intrinsic motivation and satisfaction.

While UmpleOnline demonstrates strong capabilities in areas like code generation and diagramming, user feedback indicates areas for improvement, particularly in usability and expanding functionality.

As we reflect on these findings, it is clear that addressing both technical and emotional aspects of user experience is crucial for the continued development and adoption of software

modeling tools. This study serves as a foundation for considering emotional factors in software modeling tools.

In this study we disregarded the mediator factors, but according to Peifer et al. (Peifer et al., 2022) there are several contextual factors like cultural and personal traits e.g 'autotelic personality', mediating experience of flow. Further research is needed to examine and assess these factors in the software modeling field.

Further research is needed to expand this research by including other modeling tools to facilitate comparative analysis of flow experiences. We also encourage other researchers to build upon our findings and contribute further knowledge to the domain of flow experience in software modeling.

Chapter 5 Results of Grounded Theory

This chapter presents a qualitative empirical study that aimed to understand the attitudes of software developers towards software modeling practices. Although software modeling provides acknowledged benefits, such as improving quality, facilitating superior comprehension of system behavior, and enhancing stakeholder communication, it remains significantly underutilized among developers. The content in this chapter is partly adapted from a paper published in 2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion as of October 2023 (Kalantari & Lethbridge, 2023).

This study was conducted to answer research questions RQ6 and RQ7.

5.1 Purpose of the Study

To date, many studies in the field of software engineering focus on technological aspects, largely overlooking the human factors that can significantly impact software development processes (Abrahão et al., 2017; Lenberg et al., 2015). Understanding these human factors is crucial, as software development is a cognitively intense process where developers' attitudes and beliefs can considerably influence their adoption of new tools and practices (Myers et al., 2016; Wang et al., 2012).

This study seeks to examine the deeply-rooted beliefs and attitudes that prevent developers from embracing software modeling, in addition to understanding what their current approaches and tools in development process.

Our approach to unraveling this complex issue involved conducting interviews with software developers. The complete description of data collection and analysis methodologies and procedures can be found in Section 3.4.

5.2 Results: Demographic Distribution

A total of 20 people participated in this study. The participants were from multiple geographical regions enabling us to broaden our perspective on the mindset barriers. Most

participants were from Canada and the US, but we also had participants from Australia, Germany, Italy, the UK, and Iran.

Participants' experience ranged was from 5 years to over 35 years of software development. Among our interviews we had no people we would consider early career, 8 people we would consider mid-career (5-15 years), 5 people we would consider highly experienced (16-25 years), and 7 people we would consider late career (more than 25 years). The average years of experience was 20, indicating a substantial level of expertise among our participants.

The participants in our study worked on software of numerous types, including both front and back ends, web and mobile software, as well as embedded systems. Domains included telephony, medical systems, e-commerce, fintech, government information systems, compilers, games, cybersecurity, and cryptocurrencies. They also reported using a wide variety of architectures. This diversity gave us a multi-faceted view of software modeling adoption. We ensured that we talked to developers from many aspects of the spectrum from non-modelers to modeling enthusiasts. The following shows how we would categorize these interviewees:

- Non-modelers: Developers who do not engage with modeling at all in their software development process (2 interviewees).
- Minimal modelers: Developers who utilize modeling sporadically, often for specific tasks or problem-solving (6 interviewees).
- Selective modelers: Developers who somewhat consistently incorporate modeling certain limited aspects of their work (5 interviewees).
- Modeling enthusiasts: Developers for whom modeling is an integral part of their software development process, representing the highest level of engagement with modeling (7 interviewees).

Table 11 illustrates the demographic distribution of our participants. Where possible we have described their software type in detail, but for some they just gave a general description, such as 'web/mobile', 'embedded systems', or 'backend' in general.

Table 11: Demographic Characterization of Participants

Participant	Country	Software Type	Modeling Intensity	Experience	Experience Level
P1	Canada	Web/Mobile, games	Non-modeler	5	Mid Career
P2	Iran	Web/Mobile	Non-modeler	12	Mid Career
P3	Germany	Web/Mobile	Minimal Modeler	10	Mid Career
P4	Canada	Backend, cryptocurrency	Minimal Modeler	10	Mid Career
P5	Canada	Web/Mobile, games, medical	Minimal Modeler	11	Mid Career
P6	Canada	Backend	Minimal Modeler	20	Highly Experienced
P7	US	Web/Mobile, enterprise systems	Minimal Modeler	25	Highly Experienced
P8	Canada	Embedded, compilers, operating systems, UI, graphics	Minimal Modeler	30	Late Career
P9	Canada	Distributed Systems, e-commerce, fintech	Selective Modeler	8	Mid Career
P10	Canada	Backend, consumer-facing products	Selective Modeler	8	Mid Career
P11	Canada	Web/Mobile, cloud-based	Selective Modeler	12	Mid Career
P12	US	Telephony, medical systems, video, security	Selective Modeler	26	Late Career
P13	UK	Embedded	Selective Modeler	30	Late Career
P14	US	Backend	Modeling enthusiast	20	Highly Experienced
P15	Canada	Cybersecurity	Modeling enthusiast	20	Highly Experienced
P16	Australia	System Engineering	Modeling enthusiast	22	Highly Experienced
P17	US	Web/Mobile and desktop, Government Systems	Modeling enthusiast	30	Late Career
P18	Australia	Embedded, scientific instruments, appliances	Modeling enthusiast	30	Late Career
P19	Italy	Language Engineering	Modeling enthusiast	33	Late Career
P20	US	Government Systems, health Services	Modeling enthusiast	40	Late Career

5.3 Grounded Theory Procedure

After completing our grounded theory analysis of the interview data, we identified several pivotal themes and sub-themes across the various topics discussed. To ensure objectivity in our sub-theme analysis, we employed a distinct participant counting methodology. Each sub-theme was counted only once per participant, regardless of how many times an individual mentioned it. This method mitigates the risk of a single participant's experiences disproportionately influencing the data, ensuring a more equitable representation of themes.

Additionally, this approach aids in recognizing both the shared experiences and the unique perspectives among the participants. The subsequent sections will detail each topic, presenting the associated themes and sub-themes, and will include pertinent participant quotations to illustrate these findings.

5.4 Emerged Topic: Mindset Barriers of Developers in Utilizing Modeling

Reflecting the mindset barriers to software modeling adoption among developers, several key themes and sub-themes emerged from the data; their frequencies are shown in Table 12. The wording in italics in Table 12 expresses the mindset barriers themselves. We also use colour coding for each of the top-level themes, and will reuse this colour-coding later in the thesis to help the reader make connections. A key to the codes is shown before chapter 1.

Many of the barriers reinforce others, leading to vicious circles. In this and certain subsequent tables, the 'Reinforced by' column indicates our assessment of how certain subthemes interact and reinforce one another. This brief explanation helps to highlight the interconnected nature of the subthemes and provides context for understanding their relationships.

Table 12: Emerged Themes A to E and Sub-Themes Related to Mindset Barriers

Sub-theme	Description	Reinforced by
Theme A: Tooling and Support Concerns (Section 5.4.1)		
Sub-theme A1: Excess Complexity (4 participants)	Modeling tools and processes often add a layer of complexity that hinders their adoption by giving rise to *fears of a long learning curve and frustration during use*. *Some who start to adopt are deterred from persisting.*	A2, A3, E2
Sub-theme A2: Insufficient Support (2)	There is inadequate help, examples, or documentation to assist developers in using modeling tools effectively and insufficient maintenance of some tools. *Only experts or those with motivation and time may be able to figure out aspects of modeling tools or work around bugs and persist in using them; others are deterred.*	
Sub-theme A3: Insufficient Utility (5)	Developers desire missing features or greater practical value or usefulness in modeling tools. *Real lack of certain features may lead developers to not appreciate the value of the features that are provided, or even notice them.*	
Sub-theme A4: Obsolescence (1)	Concerns about whether the tool or the models will survive or be easily upgradable. *This is a very real fear for uptake of new and niche tools.*	C1

Theme B: Challenges from Modern Work Paradigms (Section 5.4.2)		
Sub-theme B1: Emergence of Agile Approaches (6)	There is a shift towards agile methods, which emphasize rapid and flexible response to change. This approach explicitly prioritizes working code over documentation, and fostering the *perception that modeling is less useful.*	B2, B3, B4, D3
Sub-theme B2: Effective Communication Mechanisms (2)	The rise of effective alternative means of communication (like video links, collaborative commenting and editing), give developers the *perception they need less or no modeling to achieve good quality and productivity.*	
Sub-theme B3: Advancements in Documentation Techniques (2)	Methods of automatically documenting software, such as generating APIs and diagrams from code, have proved very effective. This is vastly quicker than manual after-the-fact modeling and can be kept in synch. *The value of in-advance modeling (prior to coding) therefore seems lower.*	
Sub-theme B4: Rise to Dominance of Standard Frameworks and Architectures(2)	Most teams now use standard software frameworks, which have well-understood ways of structuring files and code, as well as standard architectures, such as Docker containerization and with microservices. These tend to include their own simplified modeling approaches, typically written as YAML files. *These can render traditional, more complex modeling tools redundant, or at least give the impression that they are.*	
Theme C: Behavioral Resistance (Section 5.4.3)		
Sub-theme C1: Lack of Top Management Support(3)	The absence of endorsement or encouragement from the upper managerial level for the use of modeling tools, due to cost issues and other issues in this table causes *the adoption of modeling tools to suffer throughout the organization.*	A4, B2, B4, C2, E1
Sub-theme C2: Project Manager Resistance(1)	Project managers, often view *modeling as a waste of time, something they cannot control, and something that may not contribute to product development; they thus resist the use of modeling.*	A1, A4, all B, C1, D3, E1, E2
Sub-theme C3: Developer Resistance(3)	Developers are reluctant to adopt modeling tools, due to other factors in this table, as well as a desire for the flexibility and freedom to do things their way. *Developers hence mutually reinforce a cultural mindset against modeling that it is hard for an individual developer to break free from.*	All others
Theme D: Lack of Perceived Value (Section 5.4.4)		
Sub-theme D1: Dominance of Code(4)	Primary importance is placed on code (independent of whether agile practices are used). Developers are judged by delivered code. Since code is what executes, it is the only thing actually required to be delivered, debugged and kept up to date. *Keeping models in synch with code is perceived as excessive workload. All non-code aspects of software, including models, are perceived to be of lower value.*	Most others
Sub-theme D2: Reduced Efficacy of Complex Models(4)	There is a sentiment that as models (particularly in diagram form) become more complex, their utility and comprehensibility rapidly decrease. Thus, *models are seen as less valuable or even counterproductive when the system gets large.*	A1
Sub-theme D3: Simple Product, Simple Needs(5)	Most products under development are seen as relatively simple and not safety-critical, so their design appears *easy to comprehend, making modeling seem less relevant.*	B2, B3, B4

Theme E: Knowledge and Skill Deficiency (Section 5.4.5)		
Sub-theme E1: Lack of Awareness(3)	*People just don't know that modeling tools exist, and/or don't appreciate their power.*	
Sub-theme E2: Lack of Skills(2)	Developers are trained to code. Few are trained to model. *Learning to model appears to developers and managers to be a challenge, and hiring modelers appears not to be easy.*	A1, A2, D1

Before going through themes and sub-themes, it is important to note that the general themes are aligned with those reported earlier from our second SLR presented in Section 2.3.3. This alignment ensures consistency and coherence between the findings of this study and existing research. It is also worth mentioning that the themes were initially developed through our grounded theory in this chapter and subsequently applied to the literature, demonstrating how our findings support and expand upon existing knowledge in the field.

5.4.1 Emerged Theme A: Tooling and Support Concerns

This theme emerged as a key set of barriers impeding the adoption of software modeling among developers. This theme highlights the importance of tool quality, support availability, and the learning curve associated with software modeling tools.

The elements of this theme are, in fact very real weaknesses of modeling tools that can and should be addressed. However, the mindset barriers arise from the tendency of developers to abandon or avoid adoption out of fear of these issues, early bad experiences, or the bad experiences of others. Developers we talked to who enthusiastically support modeling also note many of the issues, but have risen beyond them, thus transcending the barriers. Such developers report real benefits despite the issues. So, in addition to fixing the issues, a way forward would be to enable enthusiasts to better spread reports of their successes with modeling.

We identified several sub-themes outlined in the following subsections.

5.4.1.1 Sub-theme A1: Excess Complexity

This speaks to the intricacies and complications experienced when navigating the tooling landscape of software modeling including both the tools themselves and the notations they support. One of our participants pointed to this issue as 'clunkiness' of tools and highlighted the unnecessary number of dialogues and tabs required to edit properties. They commented:

> "I often sit around thinking about things for a long time and then all of a sudden, I want to put it all down once I've got it straight in my head. And it sort of gets in the way of that process."

This particular aspect of user experience could be addressed by designing interfaces that cater to different stages of problem-solving (e.g. draft modes that don't enforce correctness) and ensure smoother, more intuitive user interaction.

Another interviewee mentioned confusing UI as a deterrent factor for uptake of tools saying:

> "There are a lot of complex menus and complex panels... Even if the languages are simple to use, the shell around the editor gives the impression of something extremely complex."

This individual further explained the impact of this complexity, stating:

> "When we present this tool to someone who's not a developer, they say, 'Okay, this is an IDE? I'm not a developer, why are you showing this to me?'."

This direct feedback underscores the significance of initial impressions and the potential deterrent effect of a complex user interface on would-be users. It highlights the necessity of aligning tool design with user experience and expectations. Despite the issues and complexities of tools, there are a few developers who advocate for modeling tools. The notion of mastering a tool despite its complexity emerged in the interview with one participant who suggested a work-around strategy for dealing with tooling limitations:

> "I have a way of working [enabling me to] avoid any issues. I know where the potholes are and know how to avoid them."

This statement also highlighted the gap between tool design and user experience even for tool experts.

5.4.1.2 Sub-theme A2: Insufficient Support

This further highlights the constraints and challenges experienced by users in the realm of software modeling tools. One point of contention raised by participants was insufficient documentation. this was expressed as a "real blocker" for users. One of the interviewees stated: "I think there's a lot of things that are not documented super well … or what certain features are and how they work. There's a couple of options that I've found [where] I could not really figure out what they're supposed to do, or how they affect things. Not being able to find an answer about it is very frustrating ..."

This sheds light on the struggle to fully understand and effectively utilize modeling tools due to a lack of comprehensive, user-friendly instructions or guidelines.

5.4.1.3 Sub-theme A3: Insufficient Utility

Interviewees remarked on features which they believed need to be available or improved in tools. These included code generation, collaboration capabilities, and ways of filtering models.

A participant addressed the insufficiency of tools and languages as a reason for not using UML:

> "One of the reasons that UML is not used is because they fail to model the most complicated part of the microservices architectures."

Another interviewee expresses a frustration and inefficiency at having to use multiple tools with overlapping functionality. The participant observes:

> "But the fact that we need to kind of use this other tool when there's so much similarity between MagicDraw and Reliance, to a certain point, to me seems unfortunate."

The inefficiencies are not just functional but also have an impact on the team, as the participant points out:

> "I think if we were able to do all of that within a single tool, that would save a lot of overhead of staff having to learn two different tools and try and duplicate the architecture and keep it consistent between multiple tools."

This statement underscores a pressing need for a more unified tool environment, which could result in improved productivity and less cognitive load on staff.

5.4.1.4 Sub-theme A4: Obsolescence

Several participants commented that they are using a niche tool with limited user base, heightening the risk that the vendor might not have a commercial incentive to continue support. Others commented on the challenges upgrading their tools and models from one version to another due to the effort involved. Concern about the long-term viability of models is a clear deterrent to the uptake of modeling.

5.4.2 Emerged Theme B: Challenges from Modern Work Paradigms

Our second major theme is that software is now being developed using very different techniques than when modeling (particularly UML) was touted as the key way forward to achieve quality.

Agility, collaborative communication, automated documentation, and standard frameworks have achieved many of the things that have also been the goal of modeling. This does not render modeling obsolete; data models and state models would seem still highly useful. But the new work paradigms have captured the conversation at the expense of modeling.

There is nothing wrong with these work paradigms – they are in fact highly valuable. Tool vendors must, however, find ways to integrate modeling with them.

5.4.2.1 Sub-theme B1: Emergence of Agile Approaches

The rise and widespread acceptance of agility emerged as a significant factor dissuading the utilization of modeling within our participant group. Several reasons contributed to this, including the typical characteristics of agile teams – smaller size, expedited task schedules, and an ability to adapt quickly to a rapidly changing environment.

Furthermore, the lower cost associated with rectifying mistakes in an agile framework was identified as an element of consideration within the conversations that we conducted. One participant commented:

> "What we need is clarity and we need to be able to move quickly when we're having the conversations, so we need the conversations to dictate the content and not the tool to dictate the speed of the conversation."

Another participant expressed that the demand is often for "quick and dirty" representations that can be rapidly drafted and interpreted. The participant shared:

> "The importance of these diagrams in a real-world scenario is very low ... you want something quick and dirty that you can quickly draw out somewhere."

They further highlighted that the mental capacity dedicated to detailed diagrams can be extraneous given the multitude of other considerations.

5.4.2.2 Sub-theme B2: Effective Communication Mechanisms

The quality of video conferencing has improved significantly, enabling more effective remote collaboration and discussion. Collaborative editing, reviewing and issue tracking have also now become the norm. Together these have reduced the need for extensive documentation or modeling to share and explain concepts.

In smaller development teams, as noted by several participants, communication becomes inherently easier and more direct:

> "I've always been a member of a small team, 4-5 team members. So we were close in terms of communication. So there have not been much problem."

This was confirmed by another participant:

> "I think nowadays is probably the most common approach, use some sort of jumping on a call, in Slack, or Zoom call or something like that. And I'll explain what I'm thinking about doing … And then we toss ideas around."

5.4.2.3 Sub-theme B3: Advancements in Documentation Techniques

These including the rise of tools that parse code, inline comments and annotations to generate useful documentation, including diagrams, essentially creating read-only models dynamically and reducing the need for up-front manual modeling. Developers that make their code self-documenting, also challenge the need for extensive modeling.

An interviewee noted:

> "What I have found is that after high-level design, you usually don't use modeling much at all, because modern IDEs [enable you to] build your code, [that is] fairly self-documenting."

On the other hand, one interviewee proposed an alternative perspective, stating that although there exist tools capable of generating UML diagrams from source code, this approach appears to reverse the typical sequence of events in the development process. In this scenario, the generated documentation no longer serves as a guiding blueprint for development but rather materializes as a post-development artifact.

The same participant further explains that in their opinion UML diagrams are replaced by something like code snippets:

"short lines of code that show how we should use [a] particular class or how to combine it with other classes in the same repository."

5.4.2.4 Sub-theme B4: Rise to Dominance of Standard Frameworks and Architectures

Much modern software now is written around frameworks such as Spring, Django, Flask and React. Containerization and microservices also now dominate. This has helped developers to

reduce design variability and code redundancy, hence reducing the need for models since there is less design space to explore and explain. This was raised by two of our interviewees as a factor leading them to not need modeling.

One interviewee highlighted the current shift towards more dynamic and granular systems built on microservices. They noted that the adoption of UML diagrams is largely observed among individuals from a more traditional, enterprise, legacy, on-premises, monolithic world. The interviewee highlighted the transition from monolithic structures to microservices, indicating that this shift has affected the relevance and usage of UML diagrams. The participant stated:

> "Now all the companies are breaking their monoliths into small microservices. I don't see really too many people using UML in this new microservices world, to be honest."

They cited the highly dynamic nature of microservices and the rapid changes they undergo as reasons for the decline in UML usage. In the fast-paced microservices landscape, they noted, "you don't need it, just drop it, create a new microservice in the next month or so and just plug it into the whole ecosystem." They concluded that the landscape has become, "really much more dynamic."

In another example:

> "In microservices, when a team decides to take up the microservice architecture the development team should be divided into small teams: 6-7 developers and with such a small team you rarely have enough time and resources for documentation and design and etc. and I think when the team is small enough the system [they] are developing is also small, otherwise the whole thing will break down."

5.4.3 Emerged Theme C: Behavioural Resistance

Behavioral resistance to adopting modeling practices emerged as an important factor in our interviews. This is found in three levels of organizations and is derived or reinforced by most of the other themes and subthemes we identified.

5.4.3.1 Sub-theme C1: Lack of Top Management Support

Some respondents highlighted that management teams are reluctant to pay for the tools. This is illustrated by one participant's experience:

"My last company, they were not happy with paying for Astah. So, although I asked them to, they refused, and they directed me just to use PowerPoint to try to, you know, mark things up that way."

Another participant remarked,

"the management was delighted by it [Astah] … But it just never occurred to them that it was a good idea to push forward on getting and maybe even standardizing on a modeling tool. It just didn't seem like it was in their head."

Reflecting on the challenges in obtaining organizational approval for the adoption of modeling tools, one participant noted,

"I could never get the people – the powers that be – to sign off on it. Most of the shops, if they were going to pay for anything, were paying for Microsoft products that I didn't want to use."

This statement encapsulates the often-complex decision-making process involved in tool adoption, highlighting the potential for conflict between developers' tool preferences and management's purchasing decisions. However, when given the autonomy to select tools, as was the case when the same participant started their own business, they expressed,

"that's one of those tools that is really a go-to tool for me and I think helps a lot."

This indicates the potential for increased tool adoption when developers have a greater influence over tool selection.

5.4.3.2 Sub-theme C2: Project Manager Resistance

Resistance can also be seen from a project management perspective, where concerns are around the source of power shifting:

"What I find is it gets undermined by project managers [who] don't understand it anyway. Senior project managers don't like being told a better way of doing things."

This quote highlights the resistance to modeling tools stemming from a lack of familiarity or a perceived threat to established methods and roles. The respondent further elaborates on this power dynamic and its impact on tool adoption:

"They don't like the fact that someone else has the power over the technology on a project. And then they undermine it, sidestep it, not because it doesn't work, it works. But because they don't command it."

This participant concludes with a crucial observation about the importance of managerial support for effective tool adoption:

"It needs to be done at a substantial level with the blessing of senior project management with full understanding of the technology or it will not be adopted."

This reflection underscores the importance of comprehensive understanding and endorsement from project managers to successfully integrate modeling tools into the workflow, emphasizing that resistance to adoption is not a measure of a tool's failure, but rather a barrier to its utilization.

5.4.3.3 Sub-theme C3: Developer Resistance

Resistance also occurs at the level of the developers who often see modeling as an unnecessary part of the development process. This mindset barrier is driven by all the others.

As one interviewee succinctly put it:

> "Most developers just want to write their code, and I think they'll get it right on their own without the use of a model."

Another participant remarked:

> "There's a resistance from the younger guys and gals to design; they just want to code."

This tendency, coupled with tight task schedules and misconceptions about agile development, often serves as a hurdle in convincing teams of the value of design and modeling. The participant further elaborated:

> "They go, 'well agile by nature means you don't design', and [they are] just missing the point completely."

This misinterpretation of agile methodology, wherein the iterative and flexible nature of agile is mistaken for an excuse to sidestep design, underlines a key challenge in encouraging wider acceptance of modeling practices.

Maintaining autonomy and control in the development process is a sentiment stated by another software developer:

> "We prefer to have the complete control over everything, so prefer to code ourselves."

This notion was further echoed by another participant:

> "Sometimes, maybe the visual model may sound a little too formal, so maybe people don't want to follow that."

This comment illustrates a possible barrier to the adoption of modeling practices: a perceived lack of flexibility or freedom that comes with the formality of visual models. Therefore, when introducing new tools or methodologies, it's important to consider how they might affect the degree of control developers can exercise over their work and address any concerns related to this aspect.

Lastly, it is worth noting that overall, a change in mindset and culture may be necessary to overcome this resistance at all levels – from management to project managers to developers –

to fully harness the potential benefits of modeling in software development. However, it is important to consider that in the software development community, changing ingrained habits and attitudes can be a formidable challenge. This point is succinctly captured by one of the interviewees, who noted:

> "It's very difficult to change people's perspective, especially developers who are very stubborn."

Thus, the sentiment highlights the vital role of effective change management in promoting the adoption of modeling practices. Effective change management may encompass clearly communicating the benefits, providing ample training and support, proactively addressing any concerns or resistance that may arise, and a careful consideration of the specific personal traits of developers that lead them to feel resistive.

5.4.4 Emerged Theme D: Lack of Perceived Value

The analysis of collected data revealed in general that modeling has low perceived value, making this a prominent theme, impacting the implementation and acceptance of modeling practices within software development environments.

There are three sub-themes:

5.4.4.1 Sub-theme D1: Dominance of Code

The common perception among developers that the "core value lies in the code." One interviewee articulated this point clearly, stating:

> "You don't spend too much time on it because a diagram is not too important after all."

Another point of evidence from the interviewees:

> "At some point, you'll actually pay us to write code. Right?"

This perspective suggests that time spent on diagramming is perceived to be better utilized writing and refining the code, which directly contributes to the software's functionality. Therefore, diagrams and models may be seen as auxiliary or even unnecessary tasks in light of the central role of coding.

As articulated by one of the study participants, the utilization of models within their software development process is not routine, with exceptions primarily occurring when either management or clients explicitly request these models. They revealed that they use it in less than 5% of their work:

> "…because the clients are generally not interested. They're happily engaging with us looking at it as there's a black box of stuff so it's not very often that [they want to see the] architecture design … They just need the inputs, the outputs, and kind of how it's going to interface to the rest of their system."

In another case, despite the presence of diagrams within the team, they seem to serve mainly for high-level discussions or external presentations. As one interviewee clearly stated:

> "I don't really look at them that much. They would be shown to upper levels and clients or outside the company."

indicating that the perceived value of models may not be uniform across all project stakeholders.

This prevalent perception, underscoring a "diminished valuation of modeling", was also reflected in other interviews, with participants asserting that the utilization of modeling practices does not unequivocally guarantee an enhancement in the overall quality of the development process. This sentiment presents an important perspective within the community, possibly suggesting a gap between the theoretical benefits of modeling and its practical implications in software development.

Another participant raises interesting perspectives on the importance of pragmatism in the modeling process. conveying the crucial notion that adopting a modeling technique should not merely be driven by what is considered best practice. Instead, it should be dictated by the specific needs, practicalities, and potential value of the project or organization. Furthermore, the interviewee brings up the intriguing point that industries with more regulatory constraints may necessitate more upfront diligence, but this does not necessarily translate into a significantly better or higher quality product. This suggests that the perceived value of modeling does not always correlate with increased product quality, a crucial factor to consider when weighing the necessity and implementation of various modeling practices.

Another issue revolves around the perceived disconnect between models and code, and the perceived effort required to keep both in sync. As the interviewee elaborated:

> "The number one question I get from people is, okay, I've got a model. How do I keep it in sync with the code? I get that all the time...But it's a mental blocker for a lot of people because then they feel it's two separate things."

This resistance underlines a critical challenge in promoting modeling practices, highlighting the need for more effective strategies to communicate the benefits of modeling and alleviate concerns about the perceived additional workload.

5.4.4.2 Sub-theme D2: Reduced Efficacy of Complex Models

In another dimension of the discussion, it was observed that when diagrams become overly large and complex, their utility tends to diminish significantly.

Users often perceive these intricate diagrams as ineffective, rendering them practically useless:

> "…then you ended up with a gigantic diagram and [difficulty] finding the part of the diagram that's relevant. It's kind of just information that's dumped on

the page and kind of spreading outwards from the top middle. So, finding information that way is [by] chance."

Another participant mentioned:

"If you end up with a class diagram with 100 plus classes on it and all with a relationship or huge number of components which is almost impossible to put on the wall, it will be useless."

In some cases, the effort to depict a complex system in a diagram could add more confusion than clarity. The participant shared:

"Especially with things like sequence diagrams and state charts, one thing [to] notice is that when the size of it with the complexity gets too much, it becomes really hard both to contain it and then also to explain it."

5.4.4.3 Sub-theme D3: Simple Product, Simple Needs

The fact that most software systems are relatively simple, and not safety-critical. Others are rendered simpler by modern work paradigms (theme B). These form another key factor that impacts developers' and project managers' perceptions of modeling value and its usage.

As the relative product complexity or perceived complexity diminishes, there appears to be a corresponding decrease in the perceived necessity for modeling. One interviewee made a crucial observation regarding this aspect, stating:

"We're not designing traffic control software; any error is not [that] dramatic. That's why the formalism is not as important, if somebody didn't get [it] right so we will fix it on the next iteration and will clear confusion eventually so it's not the cost of misunderstanding in our case, but if it was some mission-critical software it [would be] more critical."

Stated by another interviewee:

> "We don't deal with complicated structure, so we don't use UML or anything like that. So, the reference implementation just uses Python: very simple and easy understanding classes."

The participant's comments exemplify how the inherent complexity (or lack thereof) of a product can shape the perceived value and practical utilization of modeling in the software development process.

Another interviewee expressed:

> "… So, if it's a very simple, lightweight project, there may not be any modeling other than whiteboards."

This sentiment suggests that in certain cases, traditional modeling tools may be bypassed in favor of more spontaneous, easy-to-use approaches like whiteboarding. Whiteboarding allows teams to quickly visualize and iterate on ideas, without the perceived overhead of using modeling tools. However, while this method may provide immediate clarity during brainstorming or problem-solving sessions, it may not offer the same level of long-term documentation or precision as more structured modeling techniques.

The choice of modeling practices also appears to be determined by the complexity of the system being developed. Certain modeling methodologies, such as state machine diagrams, were seen by some interviewees to be valuable for modeling more complex situations. One interviewee admitted:

> "State machine diagrams actually are very useful in complex states. We don't use them super often, because it turns out things aren't that complicated most of the time. But when they get there, we use them."

5.4.5 Emerged Theme E: Knowledge and Skill Deficiency

This theme encompasses two sub-themes: Lack of awareness and lack of skills.

5.4.5.1 Sub-theme E1: Lack of Awareness

This relates to the deficiency in the understanding or awareness of specific modeling techniques, tools, or the potential benefits that modeling can bring to the software development process.

One respondent stated:

> "I think in some cases, one factor, is simply lack of awareness about certain techniques like domain specific languages."

'Unfamiliarity' is another term brought up by a participant as a reason for underutilizing of modeling practices. This lack of awareness can limit the range of tools and techniques developers have at their disposal, potentially constraining the quality and efficiency of the software development process. It would therefore be essential to promote educational and training initiatives to bridge this knowledge gap, empowering developers with a wider array of techniques that can enhance their productivity and the quality of their output.

5.4.5.2 Sub-theme E2: Lack of Skills

This refers to the gap in the necessary technical abilities required to effectively implement and utilize modeling techniques in practice.

An informant pointed out that the learning curve associated with mastering new notation tools can be steep, acting as a deterrent to their use; this is pointed out by one of the participants who is an experienced software professional, and has made repeated attempts to teach modeling tools across multiple companies. He observed that despite acknowledging the utility of these tools, many individuals find it difficult to invest the time and effort required to overcome the initial learning curve. As he states:

> "For a lot of them, it's the learning curve, where they see the utility, but they're not going to spend that time to really figure it out."

This perspective highlights the need for easier-to-use tools, better learning resources, more powerful tools where the benefits are compelling, or all three, to facilitate the adoption of modeling tools in the software development process.

Another interviewee noted a shift in the industry's hiring practices and addressed that as a factor contributing to less adoption of modeling: as companies such as Facebook are reportedly prioritizing a candidate's overall intellectual capacity over their formal software programming education. This shift is potentially contributing to the decline in the use of formal modeling techniques, as new hires may lack the formal training to effectively utilize these tools. The informant concluded:

> "we're not looking for software program graduates we're looking for smart people, then it can be everybody. So everyone is increasingly becoming reluctant to [follow the modeling] approach so it's naturally just disappearing."

5.5 Emerged Topic: Developers' Perceived Benefits of Modeling

In the qualitative analysis of developers' perspectives on modeling, three main *benefits* emerged: *Communication*, *Better Developing*, and *Representation of Information*. The themes and sub-themes are all supported by literature (Henderson & Salado, 2021). Table 13 summarizes the themes and sub-themes in this topic.

Table 13: Emerged Themes F to H and Sub-Themes about perceived benefits of modeling

Theme	Sub-theme	Description	Reinforced by
Emerged Theme F: Communication	Sub-theme F1: Engaged People	Current and future stakeholders including non-technical individuals, management, and team members	G2, G4
	Sub-theme F2: Purpose of Communication	Approval, engagement, and team collaboration	G4
Emerged Theme G: Better Developing	Sub-theme G1: Self-Understanding	Modeling enhances developers' comprehension of systems by providing a clearer, less ambiguous visualization that aids in both design and understanding of existing systems.	G4
	Sub-theme G2: Productivity Improvement	Productivity is enhanced by using models to expedite knowledge transfer and minimize time spent on coding and revisions.	G1, G4
	Sub-theme G3: Quality Improvement	Reliability is enhanced by using graphical modeling to raise the level of abstraction, which significantly reduces errors in the development process.	G1, G2, G4
	Sub-theme G4: Forming a Basis for Other Artifacts	Modeling could be used as a basis for other artifacts such as documentation and requirements.	G1, G2, G3
Emerged Theme H: Representation of Information	Sub-theme H1: Abstraction	Modeling offers a higher level of abstraction.	G2, G3, G4
	Sub-theme H2: Simplification	Modeling offers a simplified version of concepts and systems.	G2, G3
	Sub-theme H3: Visualization	Modeling offers a comprehensible version of concepts and systems through visualization.	G1, G4

5.5.1 Emerged Theme F: Communication

We further categorized and viewed communication from two aspects. One is concerned with the people and roles who are involved in the communication, including non-technical individuals, management, and team members. The second is the ultimate purpose of communication.

5.5.1.1 Sub-theme F1: Engaged People

Communication is pivotal. Developers often use modeling to convey complex ideas to a varied audience, including non-technical stakeholders. One developer expressed:

> "Let's say if you want to explain this [how things are flowing] to the people who are non-technical, then we can just put it on a big screen and discuss about something specifically."

Another developer described the role of modeling in communicating with top managers:

> "[It's] for people that [are] just executives or just want to understand how the system works without really knowing all the nitty gritty details."

Another, highlighted the forward-thinking approach of modeling:

> "You can do design up front to engage and find stakeholders build a better piece of software."

This underscores the proactive aspect of modeling in engaging diverse groups early in the development process.

5.5.1.2 Sub-theme F2: Purpose of Communication

The *ultimate purpose of communication* is multifaceted – engagement, approval, and team collaboration.

Developers discussed using modeling tools like Astah for sprint reviews, facilitating a rapid grasp of the work accomplished:

> "I would actually do the coding first according to the agile tasks that I had. But Astah was a great tool for showing to the team at the end of the sprint, what I had been working on and how it fit in with existing software components."

Models also serve to streamline complex feature approvals and foster collective understanding, vital for project progress and team collaboration:

> "So generally, when a feature gets a little complicated, [we] need to present to the team or higher committee, essentially get approval of designs. And I find that flows of diagrams are a lot more clear to present over long paragraphs of text."

Another facet of communication that came up in interviews was *team collaboration*. Participants described modeling as an indispensable tool for maintaining team synchronization, one developer emphasized the utility of modeling in communicating:

> "It's really almost always just to communicate, elaborate design until the team knows how to work"

The act of drawing, according to another developer, is essential for integrating new functionality, ensuring that everyone is on the same page:

> "If we're talking about introducing new functionality that wasn't pre-existing in the product and where to integrate it into the existing architecture, that's the type of discussion where we'll pull up the whiteboard and we will pull up historical documents as reference so that everyone is on the same page."

The critical nature of modeling tools in these collaborative efforts was underscored repeatedly. Developers expressed that without the visual aid of diagrams, effective communication on system flows and functionalities would be significantly hindered. One stated:

> "I don't think I would be able to have proper conversations with other engineers, without these diagrams."

Developers also emphasized the efficiency brought by these models, particularly when they need to onboard new members quickly:

> "…If I have those diagrams from that effort I've made, I can then bring other people in, share the diagrams, and I can talk with them and explain what they're about."

Another developer highlighted the collaborative advantage provided by Astah, especially in an agile context:

> "When I worked with Astah... showing stuff to my teammates, that whole effort would have been severely hampered without Astah."

5.5.2 Emerged Theme G: Better Developing

Several topics were brought up under the *Better Developing* theme, including *Self-Understanding*, *Productivity Improvement*, *Quality Improvement*, and *Forming a Basis for Other Artifacts*.

5.5.2.1 Sub-theme G1: Self-Understanding

The use of models and diagrams is not just a tool for visualization but also a means to document and communicate ideas more clearly. This preference for models stems from their ability to "make it easier to understand and less ambiguous," offering a streamlined way to convey information and reduce misinterpretation.

Modeling was found to significantly enhance developers' self-understanding of the system.

As one developer noted:

> "For me it is primarily to help understanding, like develop[ing] an understanding for the system either during design to find bugs, explore the design in a way that doesn't require code or running system. And then for pre-existing systems, they really help me a lot in understanding how system works by trying to

> build, like reverse engineer a model out of the system based on my minimal understanding of the system."

The narrative from the developers converges on the point that visual representations, as opposed to raw code, facilitate a quicker and more comprehensive understanding of software components, especially during demonstrations to stakeholders who may not be well-versed in programming:

> "It would help them [stakeholders] to get that mental picture a lot faster than if I were just talking or just showing flat file code on screen. We tried to avoid showing actual code during demos, because a lot of people who attend don't understand code anyway, and it doesn't present a lot of value."

Diagrams are known as crucial tools for digesting complex design documents. A participant reflected:

> "I think it's just very hard to digest the content of the design doc without any form of diagrams."

Clarity in the development process is also enhanced through modeling. The declarative nature of models aids in succinct communication, as one participant put it, they, "help us be more concise," which in turn, contributes to a clearer understanding of the system's design.

5.5.2.2 Sub-theme G2: Productivity Improvement

Developers expressed that modeling positively impacts *productivity* through *speed of knowledge transfer* within the team, particularly when orienting new members in the team:

> "If we have a structural architecture diagram that reflects the current state, it does speed up that knowledge transfer and bringing people up to speed on board."

Furthermore, they articulated that modeling aids in reducing the time taken to understand and code complex systems thorough the tangibility nature of models:

> "It [modeling] saves coding time and understanding time, but it's just the way I work, maybe I'm a slower developer than others. And I really need that tangibility."

Within the context of productivity, participants also shared insights from their experiences, emphasizing the impact of thorough design on efficiency and time management. One participant observed:

> "You end up rewriting it after the first time anyway, because they didn't put the flexibility in, and they didn't think about the reusability aspects that they could get if they just sit down and design."

5.5.2.3 Sub-theme G3: Quality Improvement

Quality is boosted by models as well. Raising the level of abstraction through modeling was noted to reduce errors significantly. Confirmed by a well-experienced developer:

> "I know from decades of experience, that they're [those who are only interested in coding] going to make all these mistakes that you could see if only you had some graphical modeling."

5.5.2.4 Sub-theme G4: Forming a Basis for Other Artifacts

Another role of modeling is *Forming a Basis for Other Artifacts*. As developers articulated, modeling serves as more than just a preliminary step. Two developers highlighted that modeling could form *a basis for documentation*.

The compelling nature of diagrams in conjunction with text in documents was highlighted, with developers advocating for a balance: bullet points, paired with diagrams for a fuller understanding. This dual approach not only enhances clarity but also enriches the expression of ideas, as captured by a developer's insight:

> "The best combination is always short... text combined with diagrams to add a little bit more colour to what we're trying to express."

Another praised modeling as a practical way in developing user stories and *forming the requirements*:

> "To me, it's [modeling is] a very good approach to use, so that you can actually go down and start building your user stories and use that as the basis for your requirements. So I think it's very effective."

Further reinforcing the importance of integrating information, a developer suggested the value of centralizing all related artifacts within the user interface:

> "We have a separate document attached with the diagrams explaining all this information whereas if you can put everything in centralized [form] … everything in a UI … where you can click it and get some more information … that's probably good."

5.5.3 Emerged Theme H: Representation of Information

Representation of Information is the final theme that emerged from the narratives of participants about their perceived benefits of modeling. “Abstraction”, “Simplification”, and “Visualization” were the main attributes that developers valued in modeling.

5.5.3.1 Sub-theme H1: Abstraction

Developers appreciated the *high-level perspective* modeling offered, with *three* developers stating it allows you to “raise the level of abstraction.” This was further highlighted by another developer, emphasizing the role of abstraction in the discussion stage:

> “We would like the discussion to stay a little bit high level.”

Additionally, the call for a tool that can “give you slice points of your design at various abstraction levels” underscores the need for a multiple-viewpoints capability. This suggests that developers not only value abstraction in their discussions but also seek tools that can support this approach by providing flexible views of the design at different levels of detail.

5.5.3.2 Sub-theme H2: Simplification

Simplification through modeling was praised for its ability to distill complex processes into understandable visuals. The use of tools like JetBrains and DataGrip by developers exemplifies this, transforming intricate SQL queries into digestible UML diagrams:

> “A query plan from a database, is very difficult, but the graphical version is shockingly easy to understand.”

Models clarify complex concepts, making them easier to understand and significantly reducing ambiguity. As one developer expressed, “…because they provide a more declarative approach,” models are preferred for initiating discussions. Another, highlighted this preference:

"I like models, I like pictures, I think they're good to begin discussion with."

Another developer addressed the role of modeling in improving the conciseness:

"They help us be more concise."

5.5.3.3 Sub-theme H3: Visualization

In terms of *visualization*, developers praised its role by stating "visual representation... is incredibly key." They described how visualization allows for the discernment of efficiencies, patterns, and organizational structures in a manner that code alone cannot reveal. One developer emphasized:

"You can see efficiencies and patterns... that you cannot possibly see just in code."

5.6 Emerged Topic: Developers' Communication Methods

In the ecosystem of software development, like any other ever-changing field, communication is key. We were interested in understanding the communication methods developers use, whether through software modeling or alternative approaches. So, we asked developers about their methods of communication with their colleagues. Two main *Methods* emerged: *Verbally* and *Documentation and Commenting*. Table 14 summarizes the themes and sub-themes in this topic.

Table 14: Emerged Themes I and J and Sub-Themes about Communication Methods

Theme	Sub-theme	Description
Emerged Theme I: Verbal Communication	Sub-theme I1: In-Person	Direct, face-to-face discussions allowing for immediate feedback and rich, nuanced communication.
	Sub-theme I2: Digitally	Virtual meetings and messaging that offer flexibility and convenience and is more common since Covid19 pandemic.
Emerged Theme J: Commenting and Documentation	Sub-theme J1: Code Snippet	Sharing small blocks of code for illustration or troubleshooting, offering a concrete example to convey a concept or problem.
	Sub-theme J2: Comments in the Code	Inline annotations that explain the purpose, functionality, or logic behind code segments, enhancing readability and maintainability.
	Sub-theme J3: ReadMe Files	Documentation files that provide an overview of the codebase, setup instructions, and usage guidelines for other developers.
	Sub-theme J4: Wikis	Collaboratively edited web pages that serve as a dynamic repository of knowledge for project documentation and team communication.
	Sub-theme J5: Diagrams	Visual representations of system architecture, data flow, or algorithms that help to abstract and convey complex information clearly.

5.6.1 Emerged Theme I: Verbal Communication

In the era preceding remote work trends and before the global shift prompted by COVID-19, developers commonly collaborated in shared physical spaces, engaging in lively discussions to troubleshoot issues and derive solutions. This in-person dialogue facilitated rapid idea exchange and problem-solving. In the last decade, and particularly since the onset of the pandemic there has been a transition towards increased reliance on digital communication tools, reshaping how developers coordinate and collaborate across distances. To some extent verbal communication may replace modeling or enhance it when models are discussed verbally.

Consequently, we categorized Verbal communication into two sub-themes: In-person and digitally.

5.6.1.1 Sub-theme I1: In-Person

When developers are together, they like to talk about their ideas. One developer states:

> "We mostly rely on oral communication to get some answers to our questions."

This way, they can quickly exchange ideas and figure things out on the spot.

The agile methodology, with its emphasis on collaboration and flexibility, resonates through their dialogues. A participant reflects on the culture this creates:

> "I guess [in] theory, there's a lot of talk about agile [methodology], to some extent things [are] communicated verbally."

Especially when they are trying to come up with new ideas, talking is key. One said:

> "The majority of the communication, especially when we are brainstorming, is more oral; […] maybe I propose this architecture, and then we discuss it and toss it around until we finally come up with the final diagram."

Through these narratives, it's evident that verbal communication, especially in person, is not just about exchanging information but also about building a shared understanding. It's a collective intelligence of a team, where they make something that they all have had a part in creating.

5.6.1.2 Sub-theme I2: Digitally

As mentioned earlier, with the advent of all the new technologies and tools, it is no surprise that most communication is now happening online through various apps and platforms. One developer described it like this:

> "I think nowadays […] probably the most common approach, [is to] use some sort of […] call, […] in Slack, or Zoom call or something like that. And then people rely on it a lot to […] just jump into a quick call. And I'll explain what I'm thinking about doing. And then [we] discuss it. And then we toss ideas around. And it's some sort of code review kind of thing. And this is the typical way you communicate your ideas."

This goes beyond just the meetings planned in advance; it's also about getting quick answers to those on-the-fly questions that pop up throughout the day, as another developer pointed out:

> "I myself directly communicate with someone from backend to ask questions over Slack."

This shows that if developers need to ask a quick question or get a fast update, these apps are the way to go.

Then there is how developers handle their tasks and bugs. A lot of that conversation happens in apps like JIRA (*Jira | Issue & Project Tracking Software | Atlassian*, n.d.), where everything is organized into tickets. Someone said:

> "... a lot of stuff will resolve around tickets in JIRA."

It means that these apps are not just for talking; they are also where a lot of the work actually gets done.

So, using apps to talk and work together is a big part of being a developer now. It is not just about making things easier; it has become the main way they keep in touch and keep projects moving forward. The stories from the developers show that these tools are now an essential part of their daily work life.

5.6.2 Emerged Theme J: Commenting and Documentation

In software development environments, effective communication extends beyond verbal interactions into different types of *commenting and documentation*. Not all of these involve modeling, but many of them can substitute for modeling or enhance modeling practices.

5.6.2.1 Sub-theme J1: Code Snippet

One way is through Code Snippets, which are excerpts of source code shared among developers to explain functionality or demonstrate relationships between different components of the software. A participant highlights this reliance on code for detailed understanding:

> "If you want more detailed information, we look at the source code and we expect to get kind of an explanation about what it does, what's its relationship to other classes [with] some code snippets."

Developers often rely on these snippets to quickly understand and engage with each other's work, making them an essential part of the technical dialogue.

5.6.2.2 Sub-theme J2: Comments in the Code

Documentation within the code is not merely about creating records but ensuring that knowledge is accessible and transferable. As one developer puts it:

> "We also write code comments to try to explain things and we will link back to [the] design docs."

This practice embeds the rationale directly alongside the code, creating a roadmap for future developers.

The utility of in-code documentation is echoed by another developer:

> "I would say at this level, probably the most important aspect to preserve what you did is documentation in the code. I would say it's probably the most useful or common [approach, resulting in] document[ing] through Javadoc."

This highlights the importance of comments in making the code self-explanatory.

However, it is noted that while the practice is common, it lacks formalization:

> "We also write code comments to try to explain things and we will link back to that design docs, but there's no formalization there."

In some cases, developers incorporate documentation more directly into their technical materials:

> "We use a lot of inline code, that we put in documents. So instead of describing the interface and trying to draw a diagram, we just put the interface in line in code blocks."

This approach allows for a more interactive and accurate representation of the code within documentation.

The use of specialized tools is also mentioned:

> "In [the] backend they have a software called Swagger: Documenting [the] API."

Tools like Swagger automate and facilitate the creation of documentation, especially for APIs, making them more understandable and user-friendly.

Policies for documentation are also considered:

> "We've looked at putting policies in places in certain areas of our code where we know that the logic is a little bit hard to understand, sort of the actual overall flow of how this module fits into the bigger thing. The locality is

> important. For the next person that wandered into that code and is going to make a change."

This foresight aids in maintaining the code's clarity and facilitates future modifications.

Lastly, efforts to integrate documentation practices into the development workflow are underway:

> "I'm trying to figure out how to automate it correctly. But we're trying to enforce adding things like design doc links into our code commit process, so that we always brought back [by] referencing them."

This initiative underscores a commitment to maintaining a strong link between code changes and their documentation.

5.6.2.3 Sub-theme J3: ReadMe Files

Another common way of preserving and transferring information among developers is through *ReadMe* Files – documents often found in the top directory of a repository and in many lower-level directories, serving as gateways to understand a project's codebase. They are usually written in the Markdown language allowing the use of headings and emphasis, and are displayed by tools such as Github when the developer browses to a directory. These files have the vital job of explaining what, why, and how of the project. Reflecting on their importance, a developer notes:

> "If I'm doing something that is not very obvious why we did it in the code that way? Probably we put in the ReadMe section of the project."

Moreover, the 'ReadMe' files are not just a repository for explanations but also a hub for design communication. As another developer puts it:

> "I think the most common way of communication in terms of design, is look[ing] at the ReadMe of the repository. So we expect to get a summary of the main classes and their relationships in the ReadMe document."

5.6.2.4 Sub-theme J4: Wikis

Using *Wikis* as another form of documentation is also addressed by developers. Wikis offer a more flexible and comprehensive way to document and represent the complexities of code that comments within the codebase cannot convey alone. A developer describes their utility with enthusiasm:

> "…A series of wiki pages that talk about different aspects of the overall product design and [include] high level, sub-modules, sort of intricate details, and then we would actually be able to use those hyper-references within the code to kind of go back and see."

This usage is emphasized by another developer:

> "We've actually put diagrams and content into code, sometimes the modern version of ASCII [diagrams in code comments] is hyperlinking [to] a wiki entry […] that lives entirely within our organization but [gives] immediate access to some reference that deals with that part of code."

The wiki stands as a current snapshot, an immediate and accessible source of truth about the system as it stands, and away from the challenges of tracing the history of changes:

> "...That's when the wiki comes in a little bit handy, which is that I'm not going to tell you what the history is all about. I'm just [going to] tell you what the current state is. So, [the wiki] describe[s] what the system looks like."

Another developer highlights the role of Wikis as a good alternative to external documentation:

> "...and we found a good compromise of having a totally external set of documentation. Most of the time, if you were making changes you forget about it [external documentation], because you're living in the code and it's just hard to represent something that was entirely in the code."

The statement captures a common dilemma: the difficulty in balancing the need for comprehensive external documentation with the immediacy and convenience of information within the codebase.

5.6.2.5 Sub-theme J5: Diagrams

Using *Diagrams* to communicate design and shared understanding is another communication method that a number of developers mentioned:

> "If I have to communicate something, I would use this non-formal conceptual diagram. … most of the people are happy on this level of details as long as [they] know what components [are] involved and what [is] going on the high level and that's usually enough."

This reflects the practicality with which developers approach documentation in modern, agile environments where speed and adaptability are key.

Another participant notes the benefits of having a structured architecture diagram:

> "If we have a structural architecture diagram that reflects the current state, it does speed up that knowledge transfer and bringing people up to speed on board. But do we always have it? No."

This highlights a common reality in the development process: while the ideal is to have up-to-date architectural diagrams, the fast-paced nature of the work often means that maintaining such documentation is a challenge.

Using non-formal diagrams was highlighted by another developer as a communication method:

> "There will be diagrams embedded in there [JIRA]. So you can get MIRO diagrams, which could have message sequence charts, or diagrams showing packages and communication flows between those. Those aren't terribly formal."

5.7 Emerged Topic: On-boarding Methods

Integrating new team members into a development project is a critical and complex aspect of the software lifecycle. We were interested in exploring the on-boarding methods developers use, whether through software modeling or other alternative strategies.

As a result, two main *Methods* emerged: *Pair Programming, and Using diagrams*. Table 15 summarizes the themes in this topic.

Table 15: Emerged Themes K and L about On-boarding Methods

Theme	Description
Emerged Theme K: Pair Programming	A collaborative coding approach where a team member (often a new one) works alongside another (often an experienced developer) to produce code together, cross-checking each other's edits.
Emerged Theme L: Use of Diagrams	Employing visual aids to convey the architecture and workflow of the system, helping new developers understand the bigger picture and specific interactions within the codebase.

5.7.1 Emerged Theme K: Pair Programming

Pair programming (Beck, 1999), a method where two developers work together at one workstation, or at least collaboratively editing the same document where each person's changes are instantly visible, has emerged as a cornerstone in the onboarding process. Although not

directly related to modeling, the use of pair programming can involve joint modeling or referring to models while coding. A participant in our study noted that doing this provides the newcomer with insights as an initiation ritual:

> "It's like going through the whole ritual of initiation. Let's assign you a bug and let's do pair programming."

This method is not just about solving tasks but is also an opportunity for the new developer to engage in understanding the system with the guidance of a more seasoned developer. The same participant also notes a cultural shift towards pair programming, particularly as remote work has become more prevalent:

> "I used to work in an office before, and pair programming feels a little bit intimidating when you are in person... But my feeling is that pair programming... has become more a norm…I think being remote, at least in my experience has helped."

The virtual environment seems to have lowered the barriers to pair programming, making it a more comfortable and common practice.

Another aspect of onboarding through pair programming is its role in mentoring:

> "It's a mentoring process more than sit down read a book process."

5.7.2 Emerged Theme L: Use of Diagrams

The employment of *diagrams* during onboarding is a practice rooted in the belief that visual representations can simplify complex information and create a shared understanding of the system's architecture. As one developer puts it:

> "Part of the onboarding certainly is working with the new hires to communicate about how we use MagicDraw for both the report generation as well as standard conventions that we've adopted… So making sure that we're all working consistently and using relationships in the same way, is something that we need to do as part of the onboarding."

Furthermore, the educational aspect of onboarding involves teaching new developers how to interpret various diagrams:

> "We're making sure everybody understands; this is how I read a use case diagram. That's what that means... Do you understand normalization? What are the normal forms?"

While the utility of these diagrams is acknowledged, there is also an admission that they may sometimes be outdated:

> "I think sometimes these diagrams might be a little too outdated. But we do use that as pretty much one of the main sources to kind of onboard new hires."

The integration of diagrams into practical work is also highlighted:

> "...a lot of it [on-boarding process] comes down to having conversations with developers about how does this [code] work and then if you look over here this is the kind of the architecture diagram but now let's tie it into the code that you're looking at."

In conclusion, diagrams are a valuable tool in the onboarding process, serving as both educational resources and practical aids. They are not standalone solutions but are part of a larger onboarding strategy that includes mentorship, hands-on projects, design document reviews, and

discussions that collectively smooth the transition of new developers into the team. Through diagrams, new hires can visualize the system's structure, facilitating a deeper understanding of their work and enabling them to contribute more effectively.

Additionally, several approaches have been mentioned that focus on engaging new developers directly with the codebase and involving them in active review of design documents. Providing hands-on projects has also been highlighted as an effective strategy to enhance the onboarding experience for developers, fostering a deeper understanding of the system they will be working with.

5.8 Emerged Topic: Perceived Pros and Cons of Current Approach

This section explores developers' perspectives on the advantages and disadvantages of their current working methods, related to modeling. As discussed in previous sections, we examined alternative approaches to communication and on-boarding. Here, we aim to understand the perceived benefits and drawbacks of these alternative methods. Table 16 summarizes the themes in this topic.

Table 16: Emerged Themes M and N and Sub-Themes about Perceived Pros and Cons of Current Work Approach

Theme	Sub-theme	Description
Emerged Theme M: Perceived	Sub-theme M1: Collaboration	Developers value the ease of engaging with diverse stakeholders through collaborative tools, which facilitate swift feedback and enhance project efficiency.
	Sub-theme M2: Convenience	Developers prefer high-level diagrams for their simplicity and quick understanding, and they favor off-the-shelf tools for their reliability and ease of maintenance over custom solutions.
Emerged Theme N: Perceived	Sub-theme N1: Challenges in Information Management	The ease of idea exchange in modern development is hindered by insufficient documentation, especially in Agile environments where dedicated roles for documentation have been eliminated.
	Sub-theme N2: Shift Away from Standards	The move away from established standards like UML is seen as detrimental, causing issues in design representation and cohesion.

5.8.1 Emerged Theme M: Perceived Advantages of Current Work Approach

We asked participants *'What do you like about your current approach'*. Two emerging themes came up from their answers: *Collaboration* and *Convenience*, which are mutually reinforcing, each enhancing the effectiveness of the other within the workflow. By convenience we are referring to benefits that incorporate the simplicity of their approach and their personal productivity or efficiency at getting their work done.

5.8.1.1 Sub-theme M1: Collaboration

Developers have embraced the *collaborative nature* of current processes, especially the *ease of engaging with stakeholders* from different areas of the project. One developer expresses:

> "Collaboration, and the ability to get access to stakeholders that you don't necessarily work with generally. You create [and] read the documents you request for review, but then you basically dump it into a Slack channel that basically tagged the people who you want reviews from. And you immediately get feedback from people you don't expect... And then you kind of create a collaboration of comments back and forth."

This sentiment highlights the role of facilitating tools in supporting the required collaboration among developers. The shift towards collaborative tools has also been highlighted for its ability to shorten feedback loops and improving efficiency:

> "I guess the collaborative nature [of] tools like MIRO, makes it easy to work with other people, which can be nice. Whereas, in the old-fashioned ways, tools tend to be fairly isolated, you create your diagram, and then you'd share to the review process, that's the presenting kind of a month or two's work. And then if people don't like it, it's a problem. That smaller cycle time. [It] means I think if things are going off the tracks and the problems can be addressed more quickly, so you get go less-far down a bad path."

5.8.1.2 Sub-theme M2: Convenience

In the context of *using diagrams*, there is a preference for high-level diagrams over exhaustive detailing:

> "...the practice in the field is a good practice is that you don't diagram entire database you just break it up into the parts..."

By focusing on specific components, these diagrams facilitate a quicker grasp of the system's structure, highlighting a more-efficient approach.

The use of high-level diagrams is particularly favored for their simplicity and effectiveness:

> "Easy to produce, and it's just a visual aid to communicate your point."

Moreover, there's an appreciation for the convenience and reliability of off-the-shelf tools:

> "...a lot of them are kind of off-the-shelf tools means that they can be, to some extent best maintained than in-house tools."

This reflects a growing trend towards leveraging standard, widely supported tools over custom solutions, which can be more resource-intensive to maintain.

5.8.2 Emerged Theme N: Perceived Disadvantages of Current Work Approach

The process of software development is often praised for its flexibility and adaptability, especially in agile environments. However, it is crucial to address that the same attributes that make these methodologies appealing can also introduce significant challenges, particularly in terms of *Information Management* and *Standardization and Communication*.

The sentiments expressed by developers themselves shed light on the less-favorable aspects of current practices.

5.8.2.1 Sub-theme N1: Challenges in Information Management

The ease of idea exchange in current development processes is a double-edged sword. One developer points out the difficulty that arises from the absence of formal documentation when trying to onboard new members or review previous areas of the code.

> "On one hand, it's kind of easier to communicate ideas... But the kind of lack of formalism means that if you actually want to learn something, a new part of the code base... it'll be more difficult finding information you need..."

They struggle with insufficient documentation:

> "...in my previous job, there was not any sufficient document at all, and I think it's a threat to the whole computer industry..."

One participant shed light on the scarcity of documentation in the current process, linking it to a shift in team roles within Agile and Scrum frameworks:

> "Developers usually don't like to document. There used to be some analyzers to analyze and document the business, but with agile and Scrum framework, we don't have that role anymore. We have a development team and a product owner."

5.8.2.2 Sub-theme N2: Shift Away from Standards

A significant concern is the drift away from established standards such as UML (Unified Modeling Language). One developer emphasizes:

> "The problem is […] less and less […] formalism. We should not forget there was standards for those diagrams, collectively called as UML and that is for a reason. So, using a common vocabulary, common artifacts and ontology [is for] people understanding it more easily."

This deviation from standardization is seen as a detriment to the utility of design representations.

The issue of lack of standardization becomes more pronounced in microservices architecture, where there is a need for a central authority to ensure cohesion. Another developer, explains the challenge:

> "…everything is split into smaller pieces... somebody needs to be watching for that. And if somebody's not watching for those aspects, then the pieces don't fit together with each other. That's a challenge to communicate."

5.9 Emerged Topic: Different Types of Modeling Tools

As part of our data analysis, we have categorized the modeling tools used by developers into four groups as listed below. Our initial thinking was to simply group them into informal and formal, but there are major differences in capability as we go from truly informal tools to tools that understand correct syntax, and from those to tools that can understand correct semantics, and finally to tools that use sophisticated mathematical analysis. A similar approach was used to categorize the tools discussed in the Flow survey in Section 4.2 as well as Final survey in Section 6.2.5.

Figure 26 shows a Venn diagram, illustrating these tools and their capabilities.

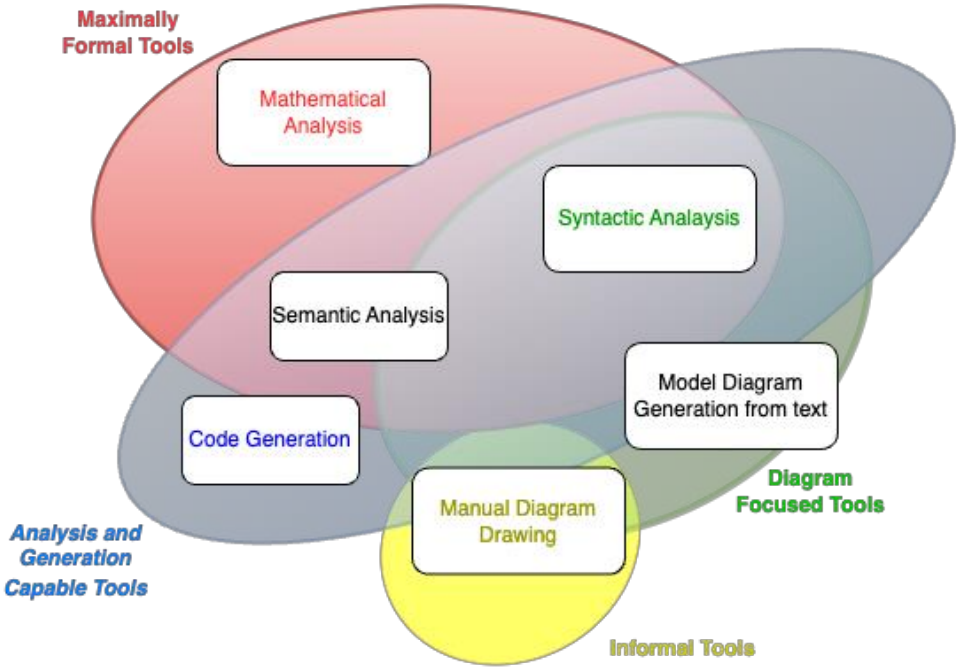


Figure 26: Categories of different modeling tools and their key capabilities

Table 17 provides detailed information on the tool categorization identified through the interviews.

Table 17: Details on tool categorization in interview

Informal Tools	**Diagram-Focused Tools**	**Analyses and Generation Capable Tools**	**Maximally Formal Tools**
Paper	Draw.io	Astah	Alloy
Whiteboard	StarUML	Rational Rose	TLA+
Virtual Whiteboard	Visio	Magic Draw	
SharePoint	Free Web-based tool	Data Grip	
Wiki Pages	Lucid Chart	JetBrains	
Google docs	Mermaid	InfosSphere	
Github	XFig		
	Data Architect		

5.9.1 *Emerged Theme: Informal modeling tools*

These tools allow drawing of diagrams of any type. They provide a palette of shapes, symbols and lines (connectors) capable of being used to represent many types of diagrams, including models. They may include special templates or setups for specific modeling languages, but they provide little or no feedback if the developer breaks the conventions of such languages in their diagrams.

Key concept: One can use them to draw many types of diagrams, not just models, and when used to draw models, you can still have other elements that don't adhere to a modeling standard.

Canonical examples: PowerPoint or even hand-drawing on paper or whiteboard

5.9.2 *Emerged Theme: Diagram-focused tools*

These have knowledge of the syntax of at least one modeling language that has a graphical representation (usually many) but may also have a textual representation. They assist the user in constructing syntactically correct models using the correct symbols, connected in a syntactically correct way (or the correct textual notation). However, these tools are not able to help detect semantic errors in models or are only able to do so minimally. Also, these tools do not provide facilities for model execution or generation of executable code.

Key concept: Good syntax checking, i.e. they can prevent you from breaking modeling language syntax, but incomplete or non-existent semantics checking

Canonical examples: Diagrams.net, Umlet

5.9.3 *Emerged Theme: Analysis and Generation Capable Tools*

These define the semantics of their models and are able to enforce adherence to the semantics by detection of errors and inconsistencies in models. They also generate outputs, such as executable code or other models. Not all such tools can generate perfect or complete code for

all modeling constructs. The semantics is typically defined using procedural or declarative code, rather than mathematical formalisms.

Key concept: Extensive semantics checking but using code for this rather than mathematics. Plus, they are capable of generating executable code.

Canonical examples: Rational Tools, Umple, Dockerfile

5.9.4 Emerged Theme: Maximally Formal tools

These use logic, set theory and other mathematical methods to define the semantics of models. They also use deduction, theorem proving, model-checking and other mathematical processes for analysis and verification of models. They may or may not be capable of generating code or other artifacts and tend to be textual.

Key concept: They define semantics of models mathematically and can analyse them mathematically too.

Canonical examples: Z, VDM, Alloy

5.10 Emerged Topic: Adoption and Usage of Modeling Practices in Organizations

In this section, we categorize and present the various quotes related to the adoption and usage of modeling practices within organizations. These quotes have been grouped into relevant themes that emerged from the participants' feedback.

5.10.1 Emerged Theme: Formality in Adoption Process

A common theme emerged among nine developers who disclosed the absence of a formal process within their organizations for utilizing modeling tools. One participant shared:

> "I found in my industrial experience so far that a lot of people don't actually use it even if they're very experienced architects."

This is confirmed in another form:

> "I've yet to be at any company where formal methods or model languages are part of the regular flow of work. It's usually something that I bring myself and use myself."

This sentiment was echoed by others, who occasionally use UML in their development process:

> "...that's not part of the process. It's just something that people create, from time to time because it's useful."

> "We don't really use UML very often. unless we're trying to get a point across.

Conversely, only one developer who was a SysML (Friedenthal et al., 2014) user, reported having a formal process in place within their organization. They affirmed, "Yeah, we do, and it's continuing to grow," suggesting a more structured approach to the utilization of modeling tools within their particular organizational framework.

5.10.2 Emerged Theme: Modeling adoption in development lifecycle

The use of modeling across various stages of software development is not uniform and varies greatly depending on individual workflows, project requirements, and organizational

cultures. In this section, we explore the diverse perspectives of practitioners on when and how they integrate modeling into their development lifecycle.

- **Early stages:**

  "So typically these are mostly done like design reviews...this would mostly be before we start the actual coding."

  "Pre-implementation, or at the very beginning of the implementation...that's when we talk the most about system flow, etc. That's when these diagrams are used the most for me."

  "Primarily in the upfront analysis and design. So we might use it to identify the 'as is' and then model that 'to be'."

  "I've used all of them...to help understand data models at the beginning stages of the project."

- **During Analysis and Design:**

  "The analysis; when we use it in the requirements definition … requirements elaboration."

  "Usually in design, I will use it if I have a design problem. I will also use it for analysis, if there is a problem, like a bug or a small enhancement that I have to make."

  "I did for a long time tend to use it in the design phase. Because I did waterfall development for a very long time."

"…So that was a very dynamic process where I would actually do the coding first according to the agile tasks that I had. But Astah was a great tool for showing to the team at the end of the sprint, what I had been working on and how it fit in with existing software components."

- **Post-Implementation:**

"…And that was sort of the reverse case where I was actually developing actively and then drawing diagrams from the code that I had written instead of mapping things out in advance."

"It's also sometimes when we find a problem; what went wrong. just recently we had an experience where we found out something was not doing what we expected so I had to go back and actually do some reverse engineering."

- **Continuous Usage:**

"I use it all the time, all day, every day, in parallel with every single thing that I do. I always have the tool running."

5.10.3 Emerged Theme: Approaches in Maintaining up-to-date models

The maintenance of software models in development practices is a critical issue that raises questions about the value and utility of modeling in the iterative and fast-paced nature of modern software development. Despite the recognized importance of keeping models synchronized with the codebase, developers often admit to neglecting this aspect.

Participants highlighted the difficulties in keeping models current and aligned with the evolving codebase.

- **Neglect in Updating**:

  "In practice, no, those diagrams never get updated, and nobody goes back and updates the document."

  "No I don't have any version control. It's really a lot of time outdated."

  "After I've created the model, I don't go back to update the diagrams."

  "There's quite a strong tendency for information to get out of date … most of the design information is not contained within those."

- **Selective Updating**:

  "When a new functionality has been introduced you have to refactor... that becomes the new content that gets updated as the code changes."

  "Certain aspects we're keeping up to date, especially with the logical data model and the physical data model."

  "I think to a certain extent, yes. But after maybe a month or two after this project has finished, I think they tend to get left behind."

- **Rigorous Updating**:

  "Absolutely, rigorously, completely, rigorously."

  "Yes, they are not just used for documentation. So they never [have] been out of sync because they are the implementation."

5.10.4 Emerged Theme: Tool Selection Criteria

When it comes to selecting tools for development work, organizations and individual developers often weigh a variety of factors. The decision-making process is influenced by both practical and strategic considerations. Developers have shed light on several metrics that play a crucial role in this selection:

- **Cost and Accessibility**

Cost is frequently a decisive factor. One developer's experience in choosing 'Astah' over other tools, highlights this:

> "Because it's free. It gave us a good trial, we could use it, we saw that we liked it. We could do it on a monthly basis. Cost is a major factor."

This sentiment is about choosing Astah, a tool that offered a favorable cost structure with the option for a monthly subscription, providing an affordable entry point for the company to assess its value.

- **Compatibility with Existing Ecosystems**

The integration within existing ecosystems is another significant metric:

"My company chose the tools... because most other companies use that and new hires will be familiar with it," explains a developer. This suggests a preference for tools that are widely recognized and used, facilitating easier onboarding and management.

- **Performance and Features**

Tools must meet the needs of the company both in terms of functionality and performance. One developer mentioned: "It is well-featured, it gave us what we wanted." Meaning they provide a comprehensive set of functionality that are crucial for the development process.

5.10.5 Emerged Theme: Experience of Code Generation

We aimed to explore the *code generation feature* of software modeling tools by gathering insights from developers about their experiences with this specific functionality.

Code generation features, while theoretically beneficial, are not widely adopted among our interviewees. Many participants expressed reluctance to use these features, citing greater appreciation for high-level abstraction rather than capabilities for implementation provided through code generation.

> "I don't personally care much about code generation 'cause I want a modeling language that's at a higher level."

> "In my own personal experience, a modeling language that can fully generate the code for you … you have to put too much detail into the model. And I don't want to be thinking about implementation details at the modeling level."

> "I don't actually use a lot of code generation."

> "I have to say I thought about it, but I've never … had a chance to examine it myself."

The reluctance to use code generation tools is often rooted in past negative experiences. For instance, one developer recounted issues with older tools like Rational Rose, where a small change led to significant problems:

> "I tried in the old days with Rational Rose and some of those older products, I found that they're very rigid. We had one project, where we made a change to base objects, we were generating everything... But we made a change to a base object, and it broke everything, and we ended up having to start over."

Another developer shared a similar sentiment after a disappointing experience:

> "It didn't do what I wanted… It just felt too cumbersome honestly, like having to generate the code just did not feel as streamlined as just doing it myself. So I never looked at it again."

Despite the skepticism, some participants acknowledged the potential utility of code generation:

> "It would be nice... to at least get bootstrapped to get your base classes organized. I think it would be helpful."

> "It actually worked very well... as long as you didn't look at what was under the hood."

> "Generating code is useful, it really kind of gives you a handful of high-level classes that you can then go in and modify and update… Once you have the basic layout of your classes and the relations between them, it can do quite a lot of work for you in a hurry."

The general consensus among developers suggests that while code generation can be a useful starting point, it often requires significant manual refinement. Developers seem to favor tools that maintain high-level abstraction and offer flexibility rather than those that attempt to automate the coding process. We do note that developers may not have experienced tools like Umple that explicitly do not require refinement of generated code.

Code generation, therefore, remains a feature with conditional benefits, heavily dependent on the specific needs and preferences of the user.

5.11 Threats to Validity

In qualitative research, validity is often addressed through the lens of trustworthiness, which includes aspects of credibility, transferability, dependability, and confirmability. We consider potential threats to validity in these contexts and discuss how we attempted to mitigate them.

Credibility: The credibility of qualitative research relates to the confidence in the truth of the findings. In our study, the primary threat to credibility might arise from the potential bias in participant responses and researcher interpretation. To mitigate this, we employed a semi-structured interview protocol, with a pre-prepared set of interview questions, but we carefully avoided injecting any bias about a particular definition of 'modeling' when we asked questions, by merely asking them how they communicate designs. In addition, peer debriefing was carried out with two researchers independently coding the data and resolving discrepancies through discussion.

Transferability: Transferability refers to the extent to which findings can be applied in other contexts or settings. Due to the specific sample of software developers in this study, the generalization of findings may be limited. We covered a broad spectrum of software types and modeling intensity, but we were only able to obtain participants from 7 countries who all happened to be male. We have provided a detailed description of the participants and the context of the research to help readers make informed judgments about transferability to their contexts.

Dependability: Dependability involves the stability of data over time and conditions. Changes in the developers' attitudes and beliefs towards software modeling over time might influence the dependability of our findings. We tried to mitigate this by conducting all interviews within a relatively short timeframe. Moreover, a detailed description of the research methodology has been provided to ensure the study can be audited or replicated.

Confirmability: Confirmability is concerned with the degree to which the findings are shaped by the respondents and not researcher bias, motivation, or interest. To increase confirmability, the data and the process of analysis have been clearly documented and can be traced back to the original data.

Despite these measures, it is important to acknowledge that no study can fully eliminate threats to validity. The findings of this study should be interpreted with these potential limitations in mind.

5.12 Conclusion

This chapter presents a comprehensive qualitative analysis of software developers' attitudes towards modeling practices, revealing a multi-dimensional understanding of perceptions that impact the adoption and usage of these tools in industry. Despite the well-documented benefits of software modeling – such as improved system quality, enhanced stakeholder communication, and better comprehension of complex systems – our study highlights that these advantages are often overshadowed by a range of practical and perceptual barriers.

At their core, the themes we uncovered reveal a challenging paradox: Despite the well-documented benefits of software modeling – such as improved system quality, enhanced stakeholder communication, and better comprehension of complex systems – our study highlights that these advantages are often overshadowed by a range of practical and perceptual barriers. As a result, developers and managers lean towards direct coding and simplified processes, where the immediate value appears to them to be more tangible and comprehensible.

In our grounded-theory study, we identified a comprehensive set of topics, themes, and sub-themes that shed light on the current state of modeling practices within organizations. Our analysis explored the perceived benefits of software modeling, as well as the barriers hindering its adoption. This detailed examination provides a nuanced understanding of how modeling is utilized, valued, and challenged across various development environments, offering insights that can guide future improvements in the integration and effectiveness of modeling tools.

Addressing the challenges related to modeling adoption necessitates a concerted effort from tool developers, educators, and industry leaders. Facilitating a shift in mindset is a fundamental strategy in addressing several of the identified barriers. Here are a few ways to inspire such changes:

- **Improvements to tools**: Vendors need to ensure their tools are simpler to use, and are more integrated with modern development paradigms such as agility and standard frameworks. Continued development of key features is important too.
- **Integration of Modeling into Workflow**: A significant barrier to adoption is the perception of modeling tools as separate from the coding process. By integrating these tools more seamlessly into the overall workflow, developers may be more likely to adopt and consistently use them.
- **Focus on the Long-term Benefits**: While the learning curve and time investment required for modeling tools can be a barrier, emphasizing the long-term benefits in marketing materials and documentation – such as easier maintenance, better scalability, and fewer errors – can help shift the mindset towards a more future-oriented view.
- **Confidence building**: Good software projects live for a long time, so modeling tools (just like compilers) need to live a long time too. This may necessitate the involvement of open-source developers, as there is concern that any commercial tool could simply cease to exist. We note that most of the modelers we reached in this study were using commercial tools.
- **Advocacy and Education**: A lack of understanding of the value and benefits of modeling tools can result in resistance to their adoption. Modeling enthusiasts should be encouraged to blog about their products and become influencers. Conducting workshops, seminars, and showcasing success stories can help to showcase the benefits of these tools, leading to a broader acceptance and adoption. This should include demonstrating how these tools can enhance productivity, quality communication, design clarity, and general problem-solving.

These strategies can contribute to a shift in the perception of modeling tools, highlighting their value and encouraging their broader adoption in the software development community.

In the next phase of our research, we will take the insights derived from this grounded theory study and test them with a larger group of participants, using statistical analysis to evaluate their validity and generalizability.

Chapter 6 Results of the Survey Regarding Perceptions of Modeling Practices and Tools

This chapter aims to elucidate the findings derived from the collected data regarding the use and perceptions of modeling practices and tools among software developers. We followed the method described in Section 3.5 and gathered a total of 86 responses. The complete survey itself is presented in Appendix 5.

This chapter provides a comprehensive analysis of the survey data, employing various statistical methods and figures to interpret the responses. The analysis covers multiple aspects, including descriptive statistics, correlation analysis, hypothesis testing, and categorization of responses to identify patterns and insights.

6.1 Data Preparation

Data preparation is a critical step to ensure the accuracy and reliability of the analysis. The survey data underwent several preprocessing steps, including data cleaning, transformation, and categorization. During this process, Likert scale responses were transformed into numerical values to facilitate statistical analysis.

In our dataset, five questions had empty cells. These were ignored in our data analysis resulting in n=85 for the five questions, one less than n=86 for the remaining questions. We considered eliminating the responses (all answers to all questions from that participant) or filling in a neutral value of 3, but we concluded the most appropriate approach was to simply allow slight variations in n. Each of the following questions had one missing value:

1. Models allow quick communication of high-level concepts, hiding details that people may not need to know or understand, or which have not yet been finalized.
2. Modeling tools can effectively analyse models and point out potential errors.
3. There is insufficient documentation and help about modeling tools and languages.
4. As tool versions change, older models become incompatible with the new versions.

5. Modeling is not very useful in a workflow where I can easily communicate with other developers through communication mechanisms like Github comments, instant messaging, collaborative editing, and video conferencing.

In addition to the Likert scale questions, responses to questions such as Experience Level and Modeling Intensity were also converted into numerical values.

Responses to several questions, as indicated in the following paragraphs, were categorized to allow comparison of responses from the various groups.

The tools mentioned by respondents were grouped into the four categories defined in Chapter 5: 'Informal Tools,' 'Analysis and Generation Capable Tools', 'Diagram Focused Tools', and 'Maximally Formal Tools.'

Respondents' countries were categorized into regions, such as Europe, North America, and others.

The domains in which respondents work, gathered from an open-ended question, were categorized into groups, including: 'Web and Mobile', 'Enterprise Applications', 'Databases and Backend', 'Data Analysis & AI', 'Safety, Embedded, and Automation', 'Security', and other specialized areas like 'Finance', 'Health', 'Telecom', 'Crypto & Blockchain', and 'Games'.

6.2 Descriptive Statistics

Descriptive statistics provide a summary of the dataset, highlighting the central tendencies and distribution of responses.

6.2.1 Demographics

The gender distribution of respondents (Question 8 in the survey) was analyzed to understand the demographic composition of the survey participants.

As Figure 27 illustrates, 85 out of 86 participants responded to this question; 71 identified as male, making up the vast majority. There were 9 respondents who identified as female and 5 who preferred not to disclose their gender. This distribution highlights a significant gender

imbalance, with a predominance of male respondents. This is presumably a result of both the actual gender imbalance in the industry, as well challenges in our sampling, and the willingness of people to respond to our request.

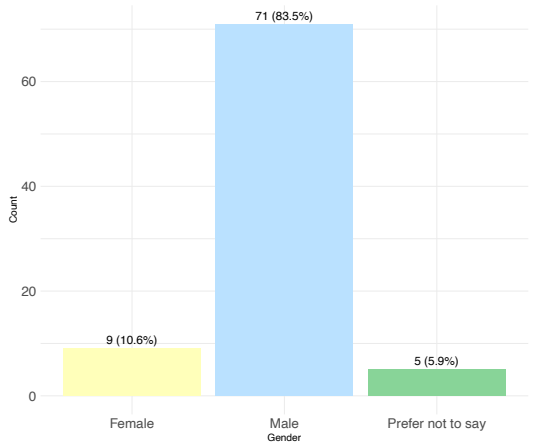


Figure 27: Gender distribution of respondents from Question 8

6.2.2 Geographic Distribution

Geographic distribution was asked in an open-ended format question (Question 6) and then analyzed and categorized into 5 regions. As depicted in Figure 28, 83 out of 86 participants responded to this question, which are shown in the figure. The majority of the respondents are from North America, followed by Europe and South America. 3 respondents did not provide an answer to this question and were excluded from the bar chart. We consider the geographic balance to be acceptable; the biggest weakness is lack of response from Asia. Part of the issue might have been firewalls in China, and cultural issues resulting in people following different types of social media which we did not target when recruiting participants.

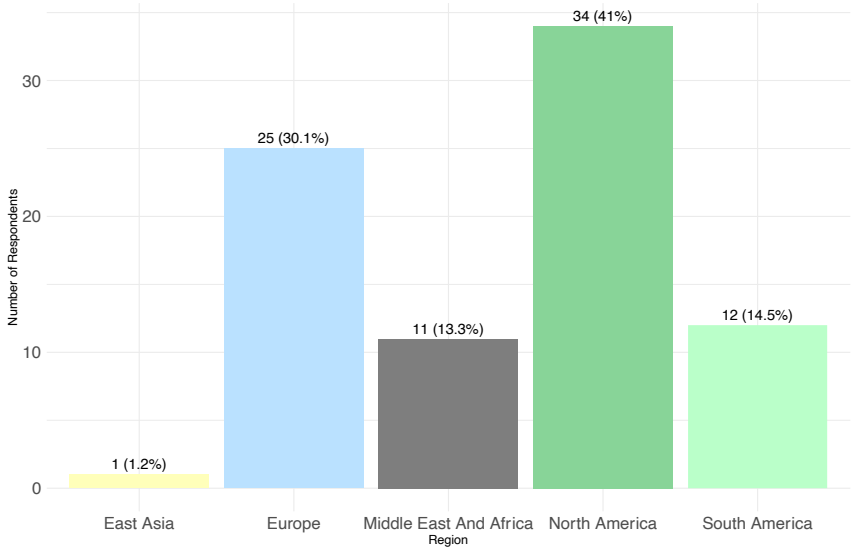


Figure 28: Geographic distribution of the respondents from Question 6

6.2.3 Modeling Intensity

Figure 29 displays the distribution of modeling intensity among respondents. This was Question 1 in the survey. We first defined modeling as, "any situation where you are creating or editing a representation of the software that is more abstract than code written in programming languages. Models may be in the form of diagrams (e.g. class diagrams, state diagrams) or test (e.g. Dockerfiles, YAML files, Textual UML variants, Formal methods language files). Models may be in specialist modeling tools, general drawing tools or even just on whiteboards or paper". We then asked each participant to categorize themselves in one of the following categories:

- Non-modeler (You do not use modeling as described above).
- Minimal modeler: (You use modeling sporadically, for specific tasks or problem solving).
- Selective modeler: (You use modeling somewhat consistently, but only for a limited set of tasks).
- Modeling enthusiast: (You use modeling extensively as an integral part of your development process).

The chart reveals that the largest group is selective modelers, followed by minimal modelers, modeling enthusiasts, and finally non-modelers. This distribution indicates varying levels of adoption and reliance on modeling practices among the respondents. The varied responses allow us to compare to see whether different subgroups had different perspectives on modeling.

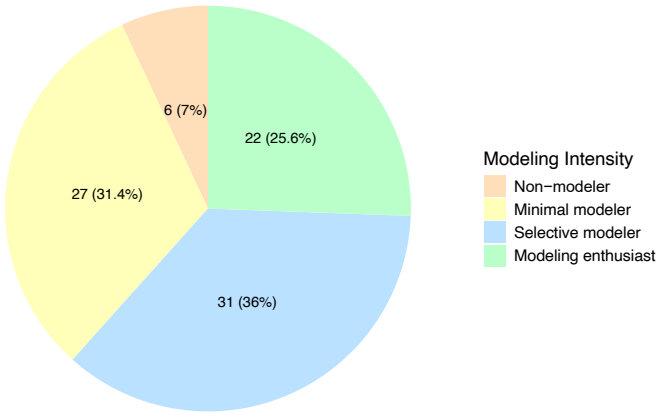


Figure 29: Modeling intensity distribution of respondents

6.2.4 Experience Level

The pie chart in Figure 30, depicts the experience level of respondents, as derived from Question 2 in the Survey. It reveals that the majority, constituting 45% of the participants, have 5-15 years of experience. This is followed by 20% who have 16-25 years of experience, 16% with more than 26 years of experience, and 12% with less than 5 years of experience. This distribution suggests that the survey sample predominantly consists of mid-level experienced professionals. Nonetheless, we have sufficient participants from other groups to do differential analysis.

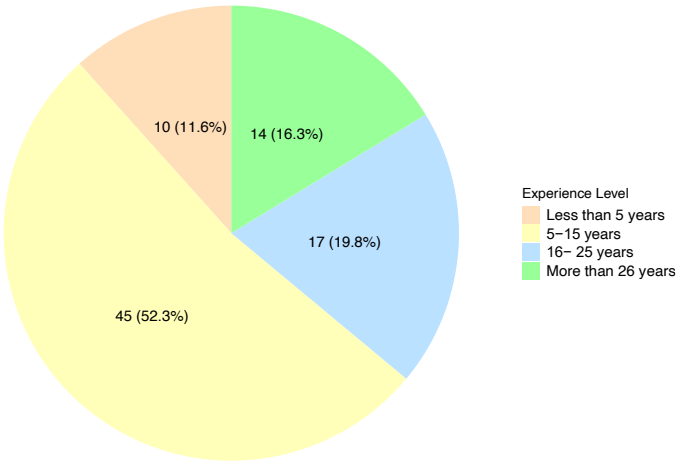


Figure 30: Experience level distribution of respondents from Question 2.

6.2.5 Tool Used

Figure 31 illustrates the distribution of tool categories used by respondents in their modeling practices. This derived from Question 5 in the survey, which listed 16 tools, from which participants could select multiple responses, followed by an open-ended "Other (please specify)" option to gather additional tools that respondents have used. The initial set of 16 tools were those that we had encountered in our grounded theory work, and from our own experience. An additional 20 tools were listed by the participants.

We studied each of the 36 tools in order to group them into the four categories we had defined in Section 5.9:

- **Informal tools**, including paper and whiteboard, where anything can be drawn, and which do not provide specific assistance for any special software modeling notation.
- **Diagram-focused tools** that support specific modeling diagrams, but do not analyse them any more deeply than pointing out syntax errors, and can not generate code.

- **Analysis and generation capable tools** that can do these tasks but not using formal logic or other mathematics.
- **Maximally formal tools** that can do mathematical analysis of models.

From the data, it is evident that Diagram-focused tools are the most frequently used category, with the highest count among all groups. This suggests that participants rely heavily on such tools for their tasks, likely due to their visual representation and ease of understanding complex concepts. They tend to be easily available, and are used outside the domain of software engineering, so they may have millions of users.

Informal tools and Analysis and generation capable tools also show significant usage with 45.3% and 40.7% of respondents respectively. The latter tend to be specialized for software engineering, and although they tend to be more powerful than diagram-focused tools, they are also more complex and have user experience challenges due to a smaller audience than diagram-focused tools.

Maximally formal tools, on the other hand, are the least used among the participants. These tools, which tend to be rigid and structured, may not be as popular due to their complexity, the need for users to understand the underlying mathematics, and the specific nature of tasks they are suited for.

Overall, the chart highlights the diverse preferences of participants in tool usage, with a clear inclination towards tools that offer visual aids and analytical capabilities, while more formal and structured tools are less favored.

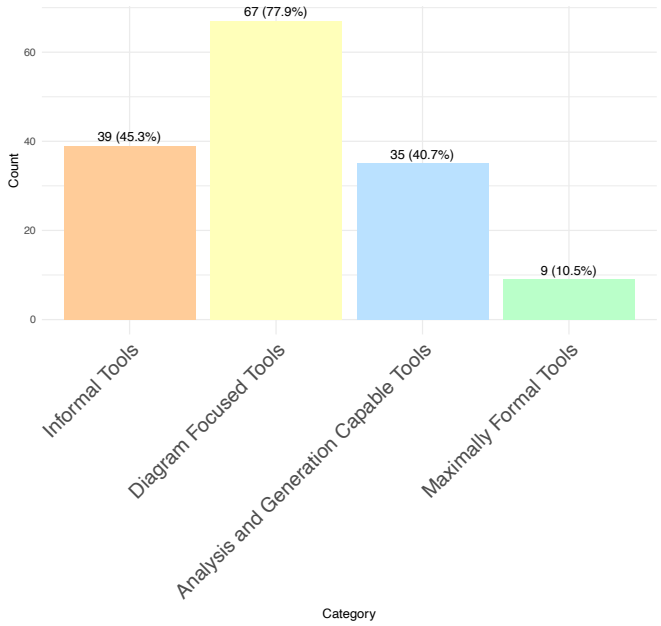


Figure 31: Tool category used by respondents from Question 5.

Table 18 provides detailed information on the tool categorization identified in the survey.

Table 18. Details on tool categorization in survey

Informal Tools	**Diagram-Focused Tools**	**Analysis and Generation Capable Tools**	**Maximally Formal Tools**
Powerpoint	ArgoUml	Umple	USE
Excalidraw	LucidChart	Papyrus	Alloy
Figma	Umlet	MagicDraw	
Balsamiq	Gliffy	Rational Software Architect/Designer	
Orca-tools	Archi /Archimate	HCL tools	
Excel	Diagrams.net /draw.io	Astah	
Paint	PlantUML	Enterprise Architect	
	Miro	DB Modelers	
	Mermaid	JAL Model Editor Compiler	
	Visio	Meta Edit	
	JUCMNav	Visual Paradigm	
	Autocad	JetBrains MPS	
	Misc diagramming	EMF	

6.2.6 *Work Domain*

Figure 32 illustrates the distribution of respondents' work domains as derived from Question 7 in the survey. The bar chart provides a clear overview of the diverse work domains represented by the respondents, emphasizing the prevalence of web/mobile and enterprise applications while also showcasing a wide range of other specialized fields.

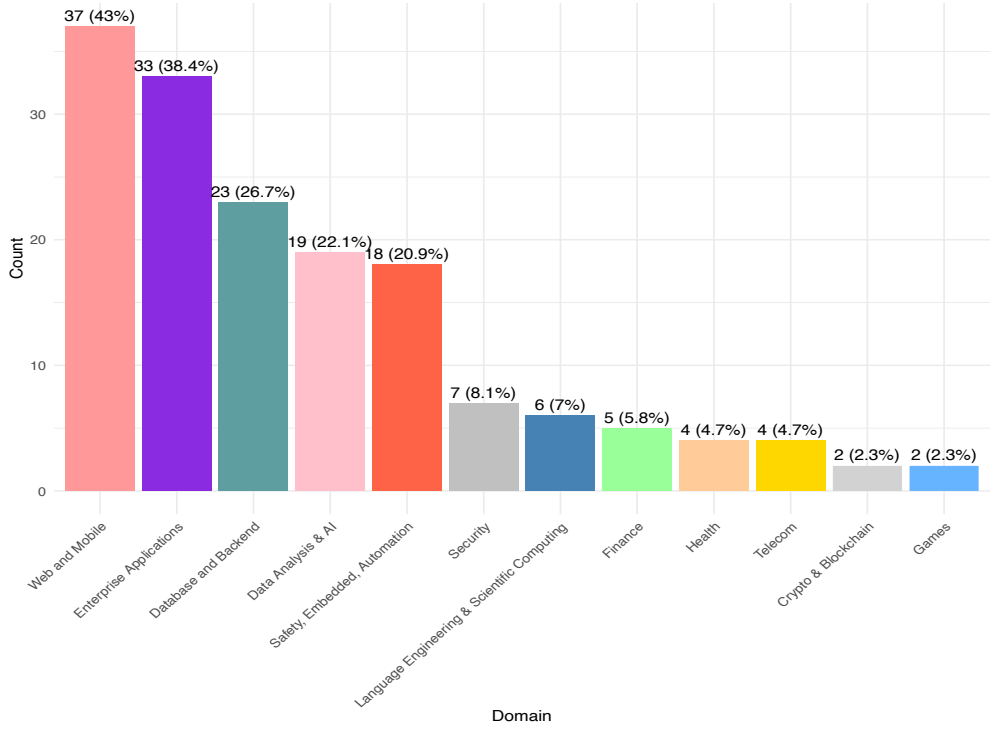


Figure 32: Work domain of respondents from Question 7

6.2.7 *Responses to Likert Scale Questions about Benefits and Challenges of Modeling*

To understand the attitudes of respondents towards modeling practices and tools, we analyzed their responses to a set of Likert scale questions. The questions were derived from our grounded theory discussed in Sections 5.4 for challenges to modeling and Section 5.5 for benefits of modeling.

The overall question posed was the following: "Thinking of your own experiences with modeling over the last year, please indicate your level of agreement with the following

statements. You can use the 'Neutral' when you either don't know enough to be able to agree or disagree, or else are partly in agreement and partly in disagreement. Remember by *modeling*, we mean any situation where you are creating or editing a representation of the software that is more abstract than code written in programming languages."

Following this heading we gave 12 statements about benefits of modeling and 32 statements about challenges, as determined from the grounded theory. Respondents were asked to respond using the standard Likert Scale, where choices were 'Strongly Agree' (which we gave a value of 5 in our analysis), 'Agree', 'Neutral', 'Disagree' and 'Strongly Disagree' (which we gave a value of 1).

The order of the 50 Likert-Scale items was randomized, with each participant seeing them in a different order. This helped reduce any bias that might have resulted from the particular sequencing of the questions.

The full questions can be found in Appendix 4. In what follows we have abbreviated the wording uses for each statement.

The results are visualized in two horizontal bar charts.

Figure 33 illustrates the distribution of responses to Likert scale questions focusing on benefits of modeling. The highest levels of agreement are seen in the following benefits:

- The communication of high-level and incomplete concepts
- Easier communication with less-technical stakeholders
- The improvement of understanding through graphic views.

These aspects are crucial as they highlight the role of modeling in facilitating clearer and more effective communication among stakeholders, which is a vital component in software development processes.

Interestingly, statements about the prevention of defects, cost reduction by early agreement, and the overall representation of software perspectives received a balanced mix of agreement and neutral responses. This suggests that while these benefits are recognized, there may be variability in their perceived impact or implementation effectiveness across different contexts or projects.

Conversely, there is notable disagreement in the statements about creating models being less bug-prone than writing code and reducing the amount of code. These indicate a potential area of concern or skepticism among respondents regarding the practical reliability and error rates associated with modeling compared to traditional coding practices.

Overall, the chart emphasizes that while modeling is widely valued for its communicative and conceptual clarity benefits, there are varied perceptions about its effectiveness in reducing errors and enhancing practical outcomes in software development.

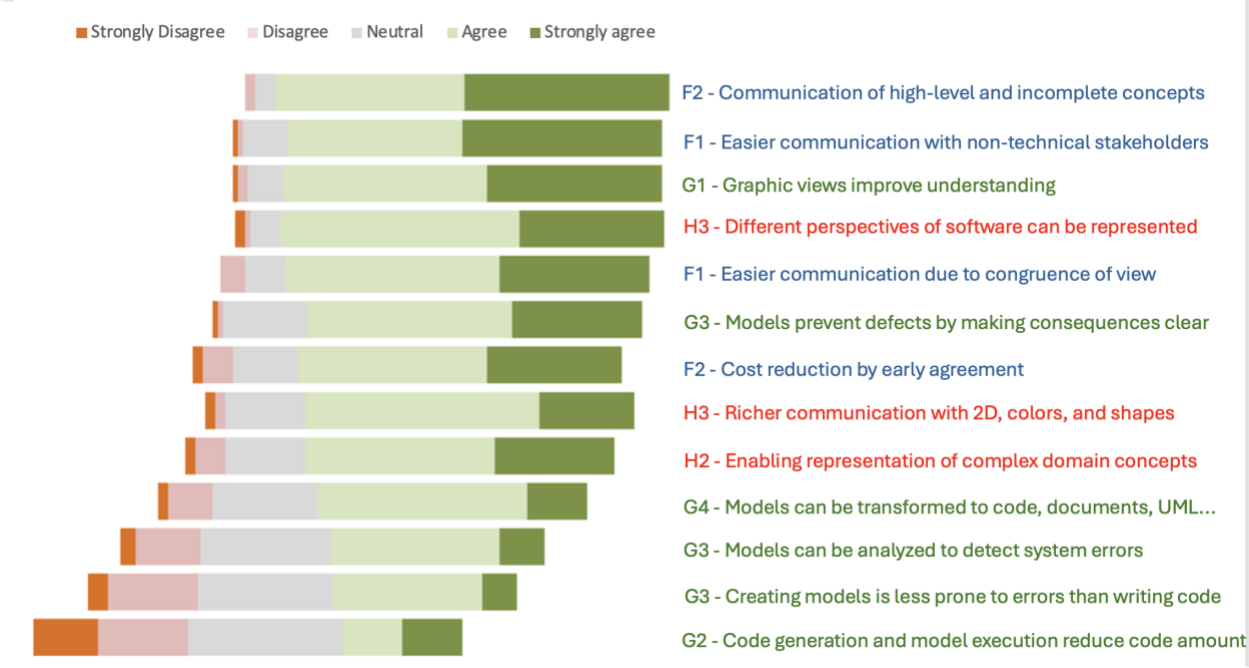


Figure 33: Responses distribution for Likert scales questions asking about benefits of modeling

Figure 34 shows Likert scale ratings regarding various challenges and drawbacks associated with software modeling.

The most significant concerns, as indicated by higher levels of agreement, include

- Lack of developer awareness of modeling benefits
- Lack of code generator quality
- Absence of a generator to update code so models get out of date.

These concerns emphasize a critical gap in awareness and technical capabilities that hinder the effective use of modeling tools.

Additionally, issues like “Lack of examples,” “Concern for obsolescence,” and “Lack of documentation” also garnered considerable agreement. This points to a perceived deficiency in resources and support that developers need to effectively implement and maintain modeling practices.

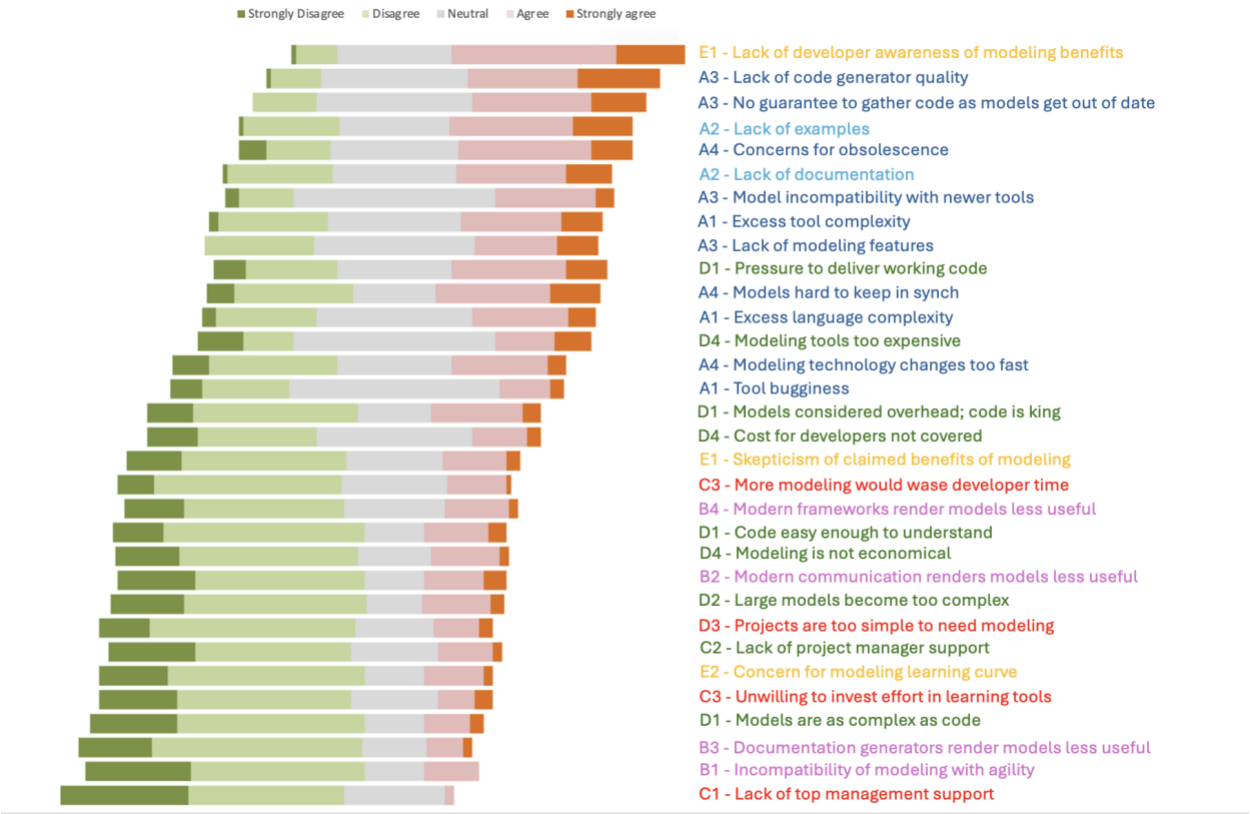


Figure 34: Responses distribution for Likert scales questions asking about barriers of modeling

6.2.8 Confidence Intervals

In this section, we use 95% confidence intervals to illustrate the perception of respondents about the benefits of modeling. Figure 35 shows the mean responses and confidence intervals for various positive statements about the benefits of software modeling. The x-axis represents the mean response on a Likert scale, and the y-axis lists the benefits of modeling.

Narrow confidence intervals suggest high precision in the mean response estimates. The colors used in the labels represent the themes identified from our grounded theory analysis in Section 5.5.

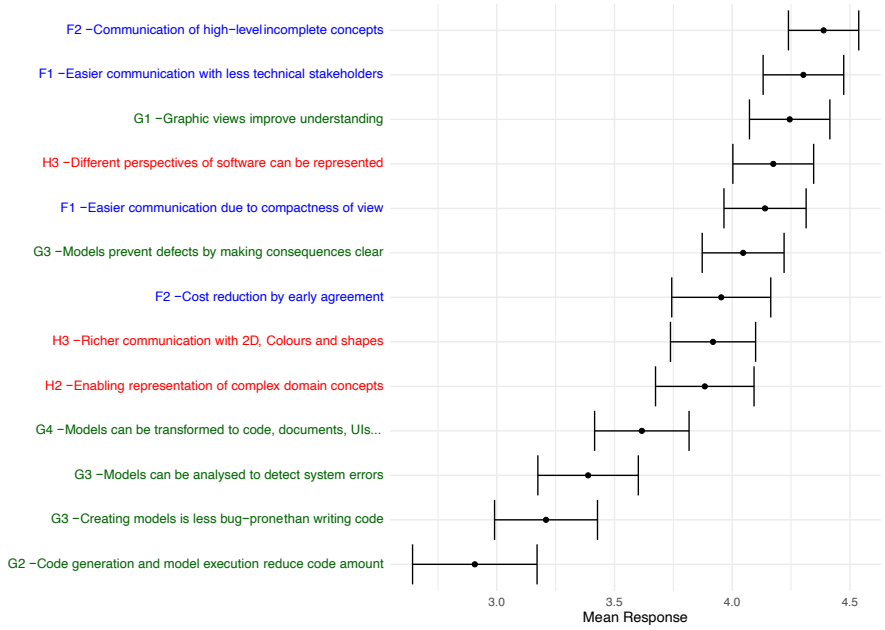


Figure 35: 95% confidence intervals of modeling benefits

Similarly, Figure 36 presents the perception of respondents about prevention factors of modeling adoption among developers. The highest mean responses are observed in the categories of "Lack of Developer Awareness of Modeling Benefits" and "Tooling and Support Concerns." Highlighting the persistent concern about lack of sufficient tooling and support to satisfy developers' needs.

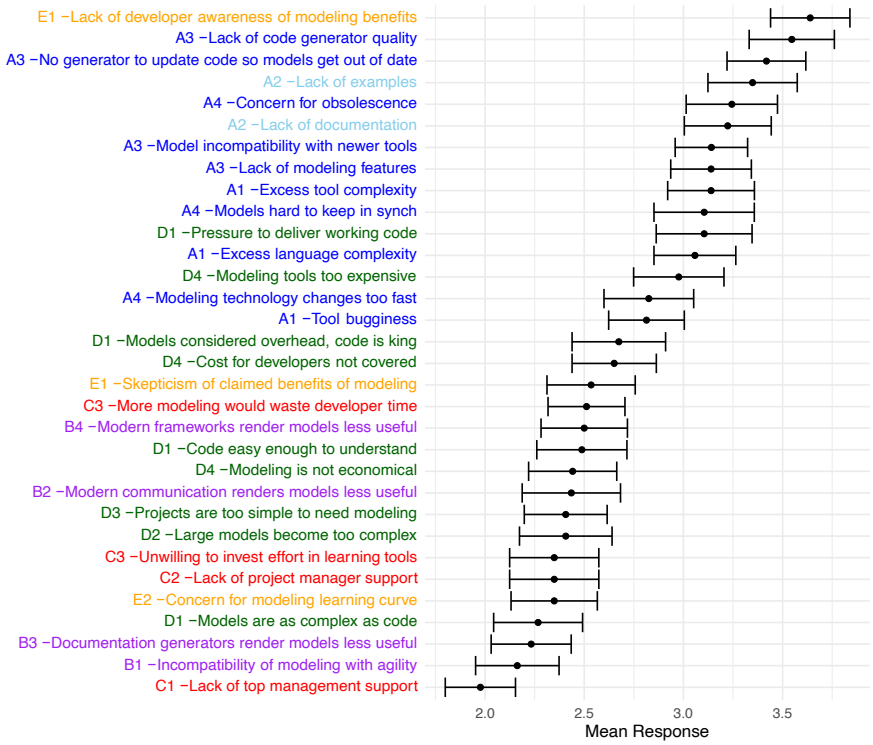


Figure 36: 95% confidence intervals of modeling drawbacks

6.2.9 Correlation Analyses in Positive Statements

The heatmap in Figure 37 illustrates the correlation between different aspects of software modeling benefits as perceived by respondents and their experience level and modeling intensity. The values within the heatmap range from -1 to 1, indicating the strength and direction of the correlations, where positive values (shaded in green) represent positive correlations and negative values (shaded in red) represent negative correlations. This visualization further categorizes questions into three distinct groups according to the themes raised from our grounded theory in Chapter 5. The categories and associated colours include Better Communication (Blue), Better Developing (Green), and Representation of Information (Red).

The heatmap reveals that both experience and modeling intensity positively influence the perception of the benefits of modeling in software development. The stronger positive

correlations with modeling intensity indicate that more extensive use of modeling is associated with a higher appreciation of its benefits.

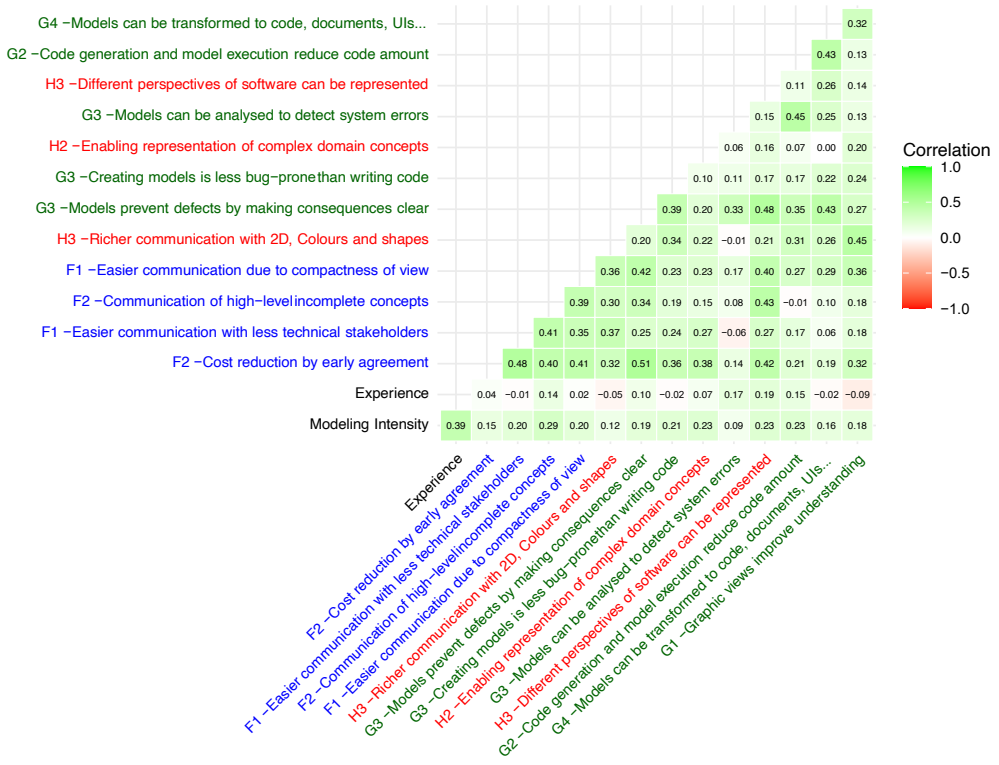


Figure 37: Pearson correlation between 'Experience Level' and 'Modeling Intensity' and Likert scales questions asking about benefits of modeling

To further asses any visible differences among different levels of modeling intensity and experience on their level of agreement to positive statement, an error bar chart was created for both different experience groups and modeling intensity groups, respectively shown in Figure 38 and Figure 39. This chart effectively highlights the general agreement on the benefits of modeling across different experience levels, with more experienced respondents often showing slightly higher appreciation for these positive aspects. Notably, modeling enthusiasts tend to rate the benefits higher, suggesting a stronger recognition of positive aspects of modeling.

To further assess any visible differences among various levels of modeling intensity and experience regarding their agreement with positive statements, an error bar chart was created for both different experience groups and modeling intensity groups, as shown in Figure 38 and Figure 39, respectively. These charts effectively highlight the general agreement on the benefits of modeling across different experience levels, with more experienced respondents often showing slightly higher appreciation for these positive aspects. Notably, modeling enthusiasts tend to rate the benefits higher, suggesting a stronger recognition of the positive aspects of modeling.

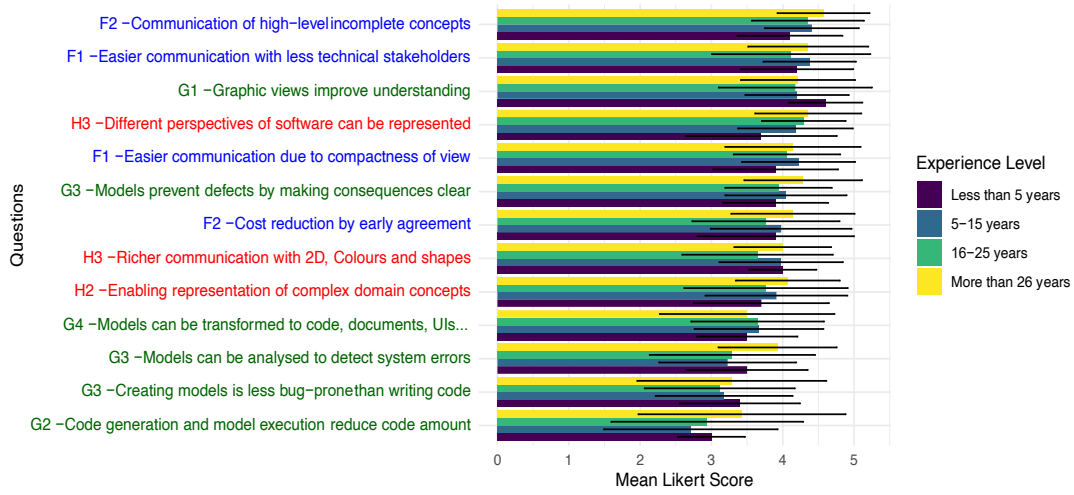


Figure 38: Error bar chart illustrating the perceived benefits of modeling across different experience levels.

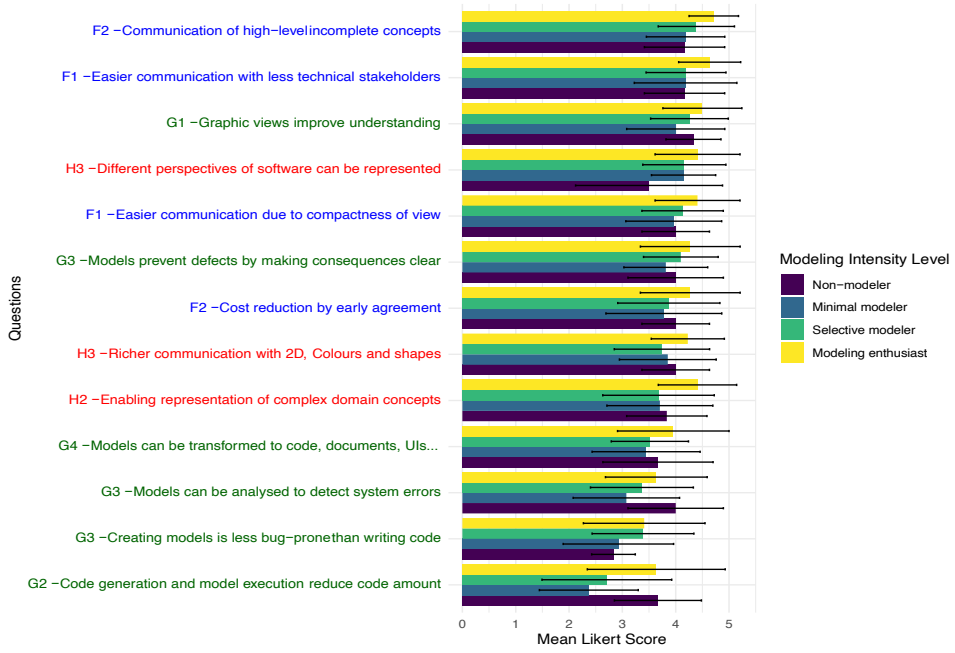


Figure 39: Error bar chart illustrating the perceived benefits of modeling across different modeling intensity levels

6.2.10 Correlation Analyses in Negative Statements

The heatmap presented in Figure 40 illustrates the correlations between various negative perceptions and challenges associated with software modeling and the factors of experience and modeling intensity. The statements are categorized into distinct groups according to the themes raised from our grounded theory in Chapter 5.

The heatmap illustrates that both experience and modeling intensity are negatively correlated with the awareness and perception of various challenges associated with software modeling. The correlation between modeling intensity and negative aspects of modeling is slightly stronger than the correlation between experience and these negative aspects. This indicates that individuals who use modeling tools more frequently are somewhat more aware of the challenges compared to those with more experience.

Two themes exhibit a stronger negative correlation compared to others: “Challenges from Modern Work Paradigms” and “Lack of Perceived Value.” Developers with lower levels of

modeling intensity tend to agree more with these categories, indicating that those who use modeling tools less frequently perceive greater challenges and see less value in modeling practices.

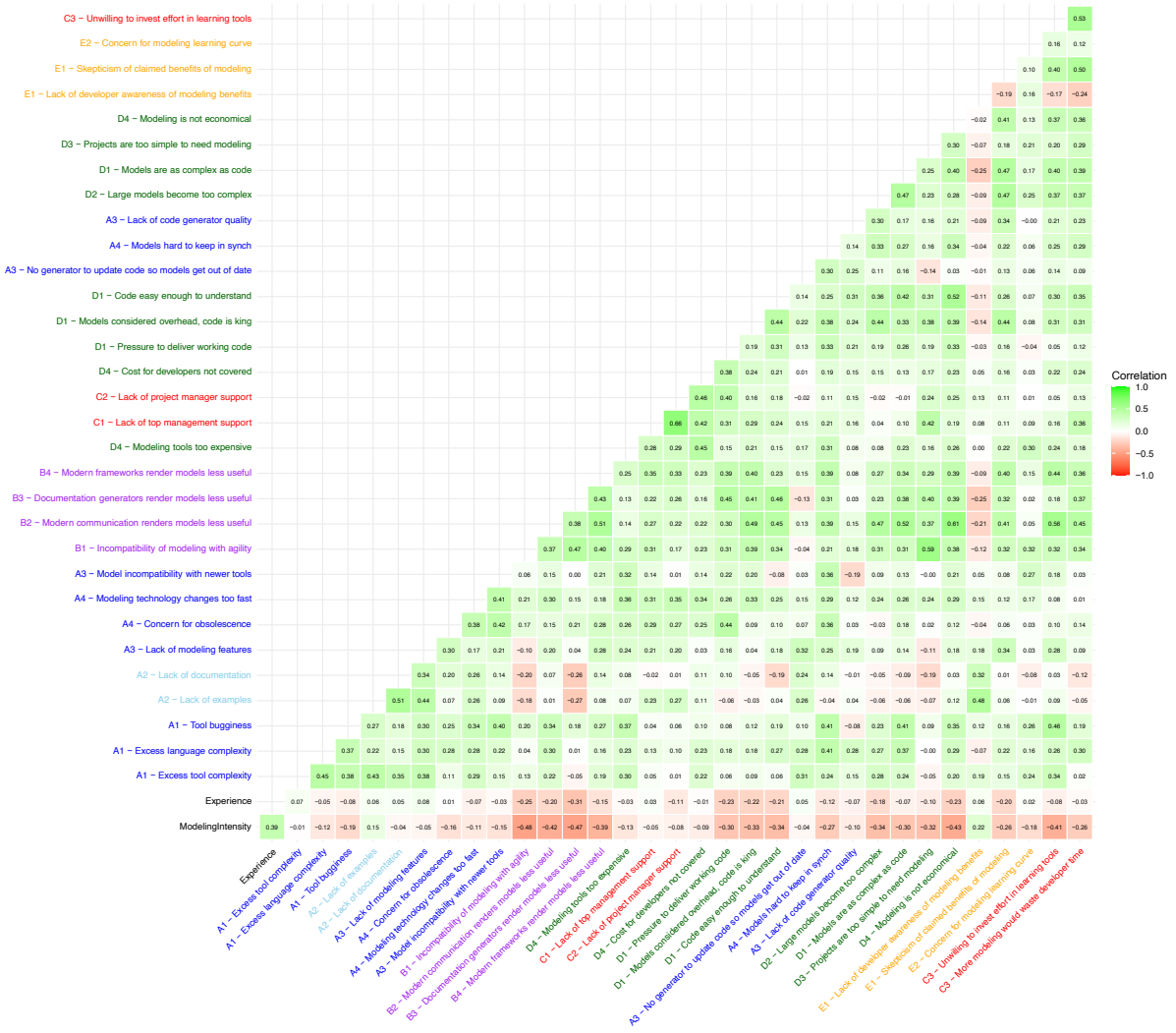


Figure 40: Pearson correlation between 'Experience Level' and 'Modeling Intensity' and Likert scales questions asking about drawbacks of modeling

Like above, we also created error bar charts for statements regarding modeling challenges in Figure 41 and Figure 42.

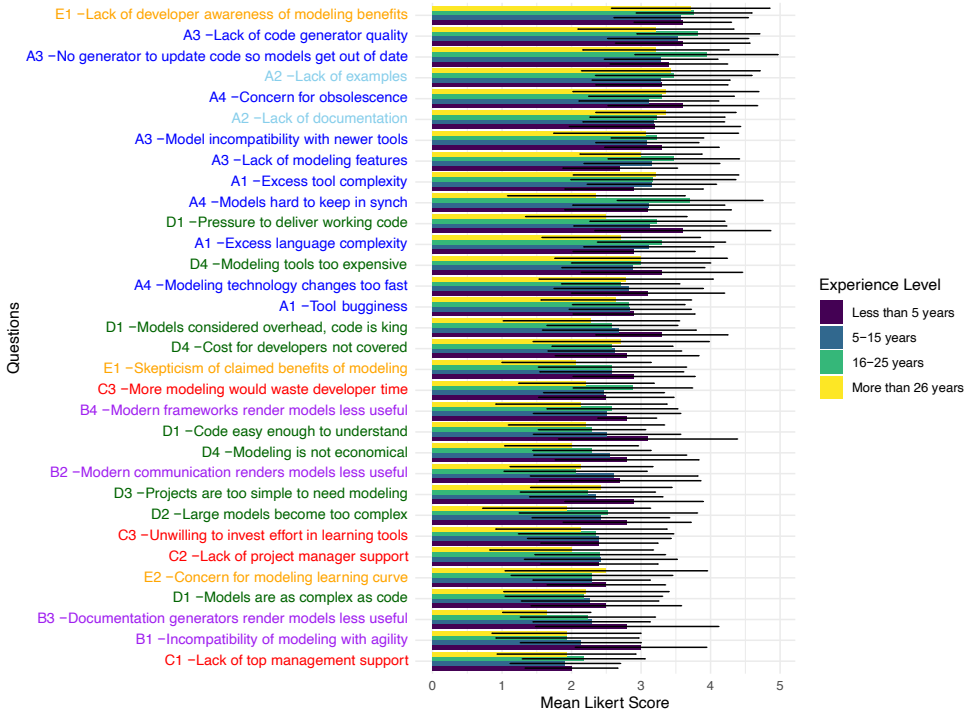


Figure 41: Error bar chart illustrating the perceived drawbacks of modeling across different experience levels.

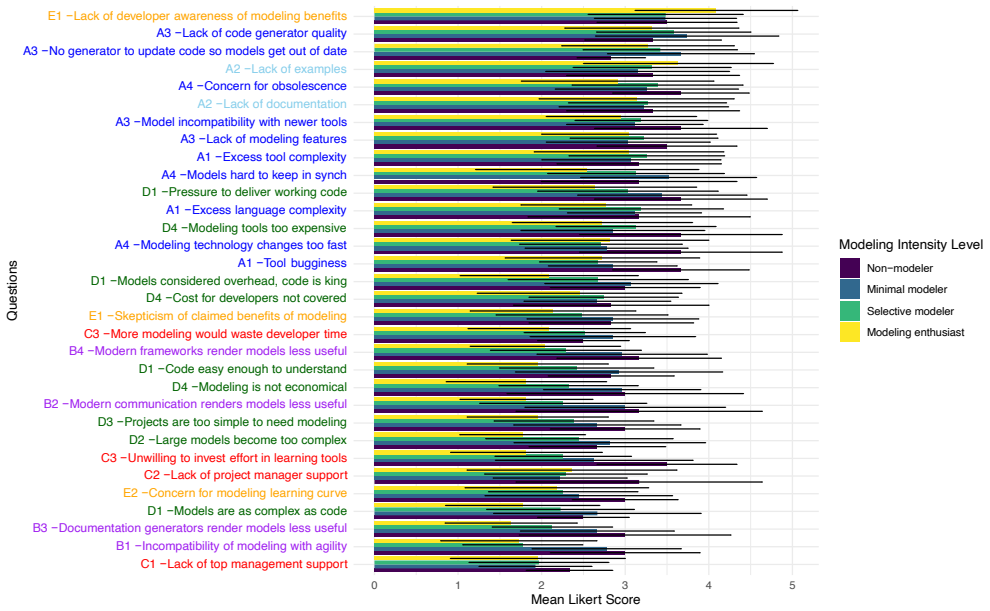


Figure 42: Error bar chart illustrating the perceived drawbacks of modeling across different modeling intensity levels

6.2.11 Hypothesis Testing

In this section, we conducted multiple ANOVA tests to examine the differences in respondents' perceptions based on multiple factors including experience levels, modeling intensity, respondents' region, the type of tool that they have used.

The Analysis of Variance (ANOVA) test is a statistical method used to determine if there are any statistically significant differences between the means of two or more independent groups. It helps to identify whether the variation between group means is larger than the variation within the groups, indicating that the group means are not all equal.

These tests were performed on both positive and negative statements regarding the benefits and challenges of modeling. The ANOVA tests aimed to identify whether there are statistically significant differences in the level of agreement among different groups.

6.2.11.1 ANOVA Analysis by Experience Levels

In order to understand how different levels of experience influence perceptions of different aspects of modeling, an ANOVA test was conducted. By examining these differences, we aim to uncover whether experience plays a role in shaping attitudes towards modeling. Based on our objective, we formulated the following hypotheses:

Null Hypothesis (H0): There is no significant difference in the responses to the positive statements about modeling across different levels of experience.

Alternative Hypothesis (H1): There is a significant difference in the responses to the positive statements about modeling across different levels of experience.

We conducted ANOVA tests for all statements, and we present only the results that indicated significant differences (indicated by a low p-value and a high F-value) in Table 19.

Table 19: ANOVA analysis results by experience level of respondents

Statement	F value	p value < 0.05
Documentation generators render models less useful	3.35	0.023
Models hard to keep in synch	3.66	0.016
Incompatibility of modeling with agility	3.23	0.026

the F-value is included in the analysis to quantify the effect size of the challenges identified in the use of software modeling tools. The F-value is ratio of the variances between the sample means of the different groups to the variances *within* those groups, a value much greater than one indicates that the means differ across these groups. If the p-value is less than 0.05, then we can reject the null hypothesis that all group means are equal.

After identifying significant differences in the ANOVA tests, post hoc analyses were performed to pinpoint the specific groups between which the differences occurred. Tukey's Honest Significant Difference (HSD) test was used for the post hoc analysis due to its effectiveness in comparing all possible pairs of means while controlling for the type I error rate.

Table 20 highlights only the statements and groups that exhibit significant differences.

Table 20: Post hoc analysis showing significant differences in responses to modeling statements among experience groups

Statement	Group1	Group2	Mean Difference	P_{tukey}
Documentation generators render models less useful	Less than 5 years	More than 26 years	1.157	0.014
Models hard to keep in synch	16 - 25 years	More than 26 years	1.349	0.007
Incompatibility of modeling with agility	16 - 25 years	Less than 5 years	-1.059	0.031
	Less than 5 years	More than 26 years	1.071	0.037

The post hoc analysis results provide deeper insights into how different levels of experience influence perceptions:

1. **Documentation Generators May be Less Useful Than Models for Those Who Model**: More experienced developers (over 26 years) find documentation generators less useful compared to those with less than 5 years of experience. This may reflect experienced developers' preference for more advanced modeling practices beyond automated documentation.
2. **Models are Harder to Keep in Sync for Those with Low Expertise:** Developers with over 26 years of experience find it easier to keep models in sync compared to those with 16-25 years of experience, possibly due to their expertise with synchronization techniques.
3. **Perception of Incompatibility of Modeling with Agility is Stronger for the Less Experienced:** Less experienced developers perceive modeling as less compatible with agile methodologies compared to their more experienced counterparts, suggesting a need for training on integrating modeling within agile practices.

6.2.11.2 ANOVA Analysis by Modeling Intensity Groups

To assess whether there are significant differences in responses to statements about modeling based on varying levels of modeling intensity, we conducted an ANOVA test across all the statements and *only* the significant results are shown in Table 21.

Null Hypothesis (H0): There is no significant difference in the responses across different modeling intensity groups.

Alternative Hypothesis (H1): There is a significant difference in the responses across different modeling intensity groups.

Table 21: ANOVA analysis results by modeling intensity of respondents (significant values only)

Statement	F value	P value < 0.05
More modeling would waste developer time	3.07	0.032
Unwilling to invest effort in learning tools	5.86	0.001
Modeling is not economical	6.813	<0.001
Projects are too simple to need modeling	3.193	0.028
Models are as complex as code	3.324	0.024
Large models become too complex	4.341	0.007
Models hard to keep in synch	2.963	0.037
Code easy enough to understand	4.042	0.010
Models considered overhead; code is king	3.759	0.014
Pressure to deliver working code	2.78	0.046
Modern frameworks render models less useful	5.367	0.002
Documentation generators render models less useful	7.82	<0.001
Modern communication renders models less useful	6.42	<0.001
Incompatibility of modeling with agility	10.88	<0.001
Code generation and model execution reduce code amount	6.280	<0.001
Enabling representation of complex domain concepts	3.12	0.030

Following the ANOVA tests, for statements showing significant differences, post-hoc tests were conducted to pinpoint the specific groups that differ from each other. The results with significant difference (P tukey <0.05) are shown in Table 22.

Table 22: Post hoc analysis showing significant differences in responses to modeling statements among modeling intensity groups (significant values only)

Statement	Group1	Group2	Mean Difference	P_{tukey}
More modeling would waste developer time	Minimal modeler	Modeling enthusiast	0.761	0.017
Unwilling to invest effort in learning tools	Minimal modeler	Modeling enthusiast	0.811	0.023
	Non-modeler	Selective modeler	1.242	0.026

Statement	Group1	Group2	Mean Difference	P_{tukey}
	Modeling enthusiast	Non-modeler	-1.682	0.002
Modeling is not economical	Minimal modeler	Modeling enthusiast	1.145	<0.001
	Modeling enthusiast	Non modeler	-1.182	0.039
Projects are too simple to need modeling	Minimal modeler	Modeling enthusiast	0.712	0.048
Models are as complex as code	Minimal modeler	Modeling enthusiast	0.894	0.014
Large models become too complex	Minimal modeler	Modeling enthusiast	1.042	0.004
Models hard to keep in synch	Minimal modeler	Modeling enthusiast	0.973	0.020
Code easy enough to understand	Minimal modeler	Modeling enthusiast	0.971	0.006
Models considered overhead; code is king	Minimal modeler	Modeling enthusiast	0.983	0.009
Modern frameworks render models less useful	Minimal modeler	Modeling enthusiast	0.918	0.006
		Selective modeler	0.673	0.041
Documentation generators render models less useful	Minimal modeler	Modeling enthusiast	1.030	<0.001
	Modeling enthusiast	Non modeler	-1.364	0.004
Modern communication renders models less useful	Minimal modeler	Modeling enthusiast	1.182	0.001
		Selective modeler	0.742	0.042
	Modeling enthusiast	Non modeler	-1.348	0.032
Incompatibility of modeling with agility	Minimal modeler	Modeling enthusiast	1.051	<0.001
		Selective modeler	1.004	<0.001
	Modeling enthusiast	Minimal modeler	1.004	<0.001
		Non modeler	-1.273	0.008
	Non modeler	Selective modeler	1.226	0.009
Code generation and model execution reduce code amount	Minimal modeler	Modeling enthusiast	-1.266	0.001
	Modeling enthusiast	Selective modeler	0.927	0.022
Enabling representation of complex domain concepts	Minimal modeler	Modeling enthusiast	-0.705	0.051
	Modeling enthusiast	Selective modeler	0.732	0.033

The ANOVA and post hoc analyses reveal significant differences in perceptions of modeling across various intensity levels. In general, modeling enthusiasts tend to view modeling more positively compared to minimal or non-modelers. They are less likely to see modeling as a waste of time, too complex, or incompatible with agility. Minimal modelers, by contrast, are more inclined to perceive modeling as unnecessary or cumbersome, especially when compared to enthusiasts.

These findings highlight that attitudes toward modeling are strongly influenced by the intensity of modeling use, suggesting that increased exposure and integration of modeling practices could shift perceptions positively. Additionally, training and education efforts could help non-modelers and minimal modelers better understand the benefits of modeling, potentially leading to broader adoption and more effective use of modeling tools.

6.2.11.3 ANOVA Analysis by Region Groups

In order to explore potential differences in responses to statements about modeling based on respondents' geographic regions (regions shown in Figure 28), we performed an ANOVA test across all the statements, and the results are shown in Table 23.

Null Hypothesis (H0): There is no significant difference in responses among different region groups.

Alternative Hypothesis (H1): There is a significant difference in responses among different region groups.

Table 23: ANOVA analysis results by region groups (significant values only)

Statement	F value	P value < 0.05
Pressure to deliver working code	3.284	0.015
Documentation generators render models less useful	5.48	<0.001
Lack of examples	2.56	0.045

For statements where the ANOVA test revealed significant differences, we conducted post-hoc tests to identify the specific regions with differing responses, and the results with significant differences are shown in Table 24.

Table 24: Post hoc analysis showing significant differences in responses to modeling statements among region groups (significant values only)

Statement	Group1	Group2	Mean Difference	P_{tukey}
Pressure to deliver working code	Europe	Middle East and Africa	-1.269	0.012
Documentation generators render models less useful	Europe	South America	-1.193	0.001
		Middle East and Africa	-1.087	0.007
		North America	-0.654	0.041

The ANOVA and post-hoc analyses indicate significant regional differences in perceptions about certain aspects of modeling:

1. **Pressure to Deliver Working Code is Greater in Certain Regions**: Respondents from the Middle East and Africa reported feeling more pressure to deliver working code compared to those from Europe (mean difference = -1.269, p = 0.012). This suggests that regional work environments or cultural factors may influence how modeling is prioritized in development workflows.
2. **Documentation Generators are Perceived to Render Models Less Useful Differently in Different Regions**: Respondents from Europe had significantly different views compared to those from South America, the Middle East and Africa, and North America. Europeans were less likely to see documentation generators as making models less useful, possibly indicating regional variations in the adoption and perceived effectiveness of modeling tools.
3. **Lack of Examples Perceived Differently in Different Regions**: While not significant P value shown in the post-hoc results, the ANOVA finding of significance suggests that different regions may experience varying levels of support and resources, such as examples and documentation, which could affect the perceived usefulness of modeling. These findings highlight the importance of considering regional differences when promoting and implementing modeling practices. Tailoring training, resources, and tools

to align with regional needs and cultural expectations could enhance the effectiveness and adoption of modeling practices across diverse geographic areas.

6.3 Threats to Validity

This study's findings are subject to several threats to validity, which could impact the reliability and generalizability of the results. These threats are categorized into internal, external, and construct validity.

- **Internal Validity**
  - **Sampling Bias:** The sample may not be fully representative of the broader population of software developers, particularly due to the reliance on specific networks for participant recruitment, which could lead to overrepresentation of certain demographics or perspectives. ***Mitigation:*** *We attempted to reduce this bias by reaching out to a diverse range of online communities and professional networks, and by encouraging participation across various geographical regions and experience levels.*

  - **Response Bias:** Participants might have provided socially desirable responses or misinterpreted survey questions, potentially skewing the data. ***Mitigation****: To address this, we ensured anonymity in the survey and emphasized that there were no right or wrong answers, encouraging honesty. Additionally, we piloted the survey to refine the wording of questions and reduce potential misunderstandings.*

- **External Validity**
  - **Generalizability:** The findings may not be generalizable to all software developers, particularly given the overrepresentation of respondents from North America and Europe, and the underrepresentation of those from Asia. ***Mitigation:*** *We acknowledge this limitation and recommend that future studies target underrepresented regions to enhance*

generalizability. Additionally, we categorized responses by region to assess regional differences and ensure that conclusions were not solely influenced by any single geographic area.

- **Construct Validity**
  - **Survey Design:** The wording and structure of the survey questions may have influenced how participants understood and answered them, affecting the accuracy of the results. ***Mitigation:*** *We conducted a pilot study to refine the survey design, ensuring clarity and reducing ambiguity in the questions. The survey was also reviewed by experts in the field to ensure that the constructs were appropriately operationalized.*

6.4 Conclusion

This chapter provided a thorough analysis of the survey data collected to explore software developers' perceptions and usage of modeling practices and tools. Using various statistical methods, including descriptive statistics, correlation analysis, hypothesis testing, and response categorization, we examined the key factors influencing the adoption and efficacy of modeling in software development.

The survey revealed important demographic insights, such as the predominance of male respondents and the significant representation of developers from North America and Europe.

The analysis showed a wide range of modeling intensities among respondents, with "selective modelers" being the largest group, followed by "minimal modelers" and "modeling enthusiasts." This variation allowed us to identify different perspectives on modeling across these groups. Additionally, the survey in our study highlighted that diagram-focused tools are the most widely used, while more formal and structured tools are less popular due to their

complexity. We should also note that, if sampling was truly random, the evidence would be strong, but there will clearly be some biases that might reduce external validity.

The responses to the Likert scale questions provided a nuanced understanding of the perceived benefits and challenges of modeling. While modeling is valued for its role in improving communication and understanding in software development, concerns remain about its practical reliability, and integration with modern development practices. The confidence interval analysis further emphasized these insights, showing varying levels of agreement among respondents regarding the benefits and drawbacks of modeling.

Our correlation and ANOVA analyses provided deeper insights into how experience levels, modeling intensity, geographic regions, and tool usage influence perceptions of modeling. The findings suggest that more experienced developers and those who engage more extensively in modeling tend to have a more positive view of its benefits, though they are also more aware of its challenges.

In summary, this chapter has shed light on the diverse perceptions and practices surrounding software modeling. These insights pave the way for further research to validate and generalize the findings, as well as for the development of strategies to address the identified barriers and enhance the adoption of modeling tools in the software development industry.

Chapter 7 Conclusion and Future Work

In this thesis, we have explored the multifaceted nature of software modeling adoption, and its perceived challenges and benefits for developers. This work was undertaken in a multi-method approach, including performing systematic literature reviews (Chapter 2), case study research involving an initial survey of users of UmpleOnline (Chapter 4), grounded theory research involving in-depth interviews (Chapter 5), and summative survey research reaching a wide range of developers (Chapter 6).

We identified many themes relating to user experience with modeling, also known as modeler experience (MX). We organized the themes into top level themes and sub-themes. These were identified initially in the grounded theory, but also were applied to and reinforced by the results of the other methods.

The most important group of themes relate to mindset *barriers*; these appear first in Table 12. We label these as themes A to E. We use a colour-coding scheme to help make the themes easier to identify in various tables throughout the thesis. These themes are:

- ***Theme A: Tooling and Support Concerns (Section 5.4.1)***
- ***Theme B: Challenges from Modern Work Paradigms (Section 5.4.2)***
- ***Theme C: Organizational and Behavioral Resistance (Section 5.4.3)***
- ***Theme D: Lack of Perceived Value (Section 5.4.4)***
- ***Theme E: Knowledge and Skill Deficiency (Section 5.4.5)***

Our most intensive analysis in this thesis related to the above five themes, but we also identified a secondary group of themes that we have labelled themes F through H relating to benefits of modeling (Table 13), themes I through J relating to communication more generally (Table 14), themes K and L relating to on-barding perhaps using models (Table 15), and themes M and N related to what developers think about their current work approach (Table 16).

- ***Theme F: Communication (Section 5.5.1)***
- ***Theme G: Better Developing (Section 5.5.2)***
- ***Theme H: Representation of Information (Section 5.5.3)***

- *Theme I: Verbal Communication (Section 5.6.1)*
- *Theme J Commenting and Documentation (Section 5.6.2)*
- *Theme K: Pair Programming (Section 5.7.1)*
- *Theme L: Use of Diagrams (Section 5.7.2)*
- *Theme M: Perceived Advantages of Current Work Approach (Section 5.8.1)*
- *Theme N: Perceived Disadvantages of Current Work Approach (Section 5.8.2)*

Additionally, we identified some tertiary themes relating to a categorization of types of modeling tools (Section 5.9) and adoption of modeling practices (Section 5.10).

In this concluding chapter, we first (Section 7.1) connect the primary themes back to the literature we identified in Chapter 2. Then show how these themes appear in all of our research methods (Section 7.2). Following that, we map the themes to the unified theory of acceptance and use of technology (UTAUT) (Section 7.3). After this, we show how our original research questions have been answered (Section 7.4), summarize our contributions (Section 7.5), outline the threats to validity (Section 7.6), suggest future work (Section 7.7) and provide some actionable recommendations for improvements for different stakeholders including managers, developers and tool vendors (Section 7.9).

7.1 Revisiting the Papers from Systematic Literature Reviews: Mapping To Our Themes

To enhance our understanding of the factors influencing the adoption of software modeling tools, we revisit the papers discussed in our first systematic literature review (presented in Section 2.2, with the list of papers with challenges found in Table 4) and map the papers and their identified issues to the adoption sub-themes identified in this thesis. Table 25 presents this alignment. We use the same colour-coding that we used earlier in the thesis, to help the reader draw connections.

Table 25: Mapping Found MX Issues from Literature to Our Thematic Framework

Paper	MX Issues	Addressed Adoption Sub-themes
(Ozkaya & Erata, 2020)	**Utility (tool):** Analyzing models, Transforming models, Managing models, Separation of concerns **Utility (language):** Extendibility, Developing formal modeling languages **Usability (language):** Complexity	**Insufficient Utility**
(Planas & Cabot, 2020)	**Usability (tool):** Difficult in use, Lack of a proper guidance **Reliability:** Bugginess	**Insufficient General Usability** **Insufficient Support** **Insufficient Reliability**
(Ren et al., 2020)	**Utility (tool):** Collaborative modeling feature	**Insufficient Utility**
(Bucchiarone et al., 2020)	**Usability (tool):** Complexity	**Excess Complexity**
(Ozkaya, 2019)	**Utility (tool):** Multiple viewpoints, Scalability, Formal verification, Round-trip engineering, Exportation, Collaborative modeling, Scripting, Project management **Usability (tool):** Complexity	**Excess Complexity** **Insufficient Utility**
(Agner et al., 2019)	**Utility (tool):** Feedback, Interaction with other tools **Usability (tool):** Complexity, Slowness, Difficult to use **Reliability:** Bugginess	**Excess Complexity** **Insufficient Utility** **Insufficient General Usability** **Insufficient Reliability**
(Stegmaier et al., 2019)	**Utility (tool):** Free sketching feature, highlighting feature, template feature **Usability (tool):** User preference in texts, shapes, sizes, colors, Undo functionality **Reliability:** bugginess	**Insufficient Utility** **Insufficient General Usability** **Insufficient Reliability**
(Savary-Leblanc, 2019)	**Usability (tool):** Complexity	**Excess Complexity**
(Badreddin et al., 2018)	**Utility (tool):** Executable artifact, Collaboration and Communication **Usability (tool):** Learning curve, Complexity	**Excess Complexity** **Insufficient Utility**
(Robles Luna et al., 2018)	**Utility (tool):** Metamodel support, Traceability and Debuggability, Monitoring technologies, Architecture modeling, Technological aspects modeling such as DB connection **Marketing:** Community and tutorials, Expensive licensing, Close source which leads to lack of control	**Insufficient Utility** **Insufficient Support** **Cost and ROI Uncertainty**
(Mert Ozkaya & Ferhat Erata, 2018)	**Utility (tool):** Various viewpoints, Sub-diagramming for the connectors (e.g., class associations), Formal verification, Collaboration support	**Insufficient Utility**

Paper	MX Issues	Addressed Adoption Sub-themes
(Pourali, 2018)	**Usability (tool):** Retainability, Error avoidance	**Insufficient Support** **Insufficient Reliability**
(Störrle, 2018)	**Utility (tool):** Layers feature **Usability (tool):** Complexity	**Excess Complexity** **Insufficient Utility**
(Agt-Rickauer et al., 2018)	**Usability (tool):** Knowledge intensive	**Excess Complexity**
(Akdur et al., 2018)	**Utility (tool):** Synchronization between software artifacts, Version management, Integration of legacy code, Model checking, Traceability and compatibility, Code generation **Usability (tool):** Need training	**Insufficient Utility** **Lack of Skills**
(Liebel et al., 2018)	**Utility (tool):** Interoperability between MBE tools, Variability management, Version management **Usability (tool):** Need training	**Insufficient Utility** **Lack of Skills**
(Agner & Lethbridge, 2017)	**Utility (tool):** Interaction with other tools, Supporting modeling aspects **Usability (tool):** Complexity	**Insufficient Utility** **Excess Complexity**
(Whittle et al., 2017)	**Utility (tool):** Model analysis, Scalability, Refactoring models, Sketching models, Tool versioning, Flexibility, Customizability, Integration, Migration **Usability (tool):** Complexity, Not match human mental model, Productivity **Usability (language):** Complexity **Emotional:** Trust **Marketing:** Cost of tools, Sustainability	**Insufficient Utility** **Excess Complexity** **Insufficient General Usability** **Obsolescence** **Cost and ROI Uncertainty**
(Vogelsang et al., 2017)	**Utility (tool):** Incompatibility, Traceability, Modularization, Testing **Usability (tool):** Learnability, Complexity **Marketing:** ROI uncertainty	**Insufficient Utility** **Excess Complexity** **Cost and ROI Uncertainty**
(Bordeleau et al., 2017)	**Utility (tool):** Collaboration facilities and customization, Graphical and textual support in a tool, Model diff/merge, Model review, Document generation. **Usability (tool):** Usability issues	**Insufficient Utility** **Insufficient General Usability**
(Ribeiro et al., 2016)	**Utility (tool):** Validation Support, Transformation support **Usability (tool):** Learnability issues	**Insufficient Utility** **Excess Complexity**
(Safdar et al., 2015)	**Usability (tool):** Learnability issues, Productivity issues	**Excess Complexity**
(Ahmar et al., 2015)	**Usability (tool):** Readability issues	**Insufficient General Usability**
(Ferreira et al., 2014)	**Utility (tool):** Visibility, Partial semantic verification **Usability (tool):** Hard mental operation Reliability: Error proneness	**Insufficient Utility** **Excess Complexity** **Insufficient General Usability** **Insufficient Reliability**

Paper	MX Issues	Addressed Adoption Sub-themes
(Williams et al., 2014)	**Marketing:** Lack of rich community	**Insufficient Support**
(Whittle et al., 2013)	**Usability (tool):** HCI concerns **Marketing:** Lack of communities	**Insufficient General Usability** **Insufficient Support**
(Condori-Fernández et al., 2013)	**Utility (tool):** Customizability **Usability (tool):** Novice users are not guided, Interfaces are not visually consistent, Some interfaces provide too much information with regard to the available space, Error messages do not help the user to solve the mistake, Icons are not self-explicative, Some interface titles are confusing, The tool does not provide undo and redo facilities, Some elements do not provide a menu when the user clicks with the right button of the mouse, Some interfaces are obtrusive (they do not allow showing the window below), Some functions are only reachable by means of icons, but not through the menu	**Insufficient Utility** **Insufficient Support** **Insufficient General Usability**
(Kuhn et al., 2012)	**Utility (tool):** Model diffing, Point-to-point traceability, **Utility (language):** Lack of problem-specific visual "little languages" **Usability (tool):** Long build-cycles prevent live modeling	**Insufficient Utility** **Reduced Efficacy**
(Heena & Ranjna, 2011)	**Utility (tool):** Round-trip engineering, All UML diagrams, UML 2.0, Document generation	**Insufficient Utility**
(Panach et al., 2011)	**Utility (tool):** Compatibility issues **Usability (tool):** Guidance, Error management, Consistency and Adaptability violations, User control	**Insufficient Utility** **Insufficient Support** **Insufficient General Usability**
(Huang et al., 2011)	**Utility (tool):** Shortcomings in terms of supported functionality **Usability (tool):** Not support different levels of expertise	**Insufficient Utility** **Insufficient Support**
(Eichelberger et al., 2009)	**Utility (tool):** Compatibility problems	**Insufficient Utility**
(Saraiva & Silva, 2008)	**Utility (tool):** Model transformations **Utility (language):** Support for language specification (syntax and semantics) **Usability (tool):** Complexity	**Excess Complexity** **Insufficient Utility**
(Auer et al., 2007)	**Usability (tool):** Complexity	**Excess Complexity**
(Störrle, 2007)	**Utility (tool):** Version control, Modularity, Migration, Releases, Backups and deployment	**Insufficient Utility**

Paper	MX Issues	Addressed Adoption Sub-themes
(Egyed, 2006)	**Utility (tool):** Consistency checking	**Insufficient Utility**
(Lahtinen & Peltonen, 2005)	**Usability (tool):** Complexity	**Excess Complexity**
(Seffah & Rilling, 2001)	**Usability (tool):** Interface personalization, Retainability, Error avoidance, Speaking the developer's language, Keeping the developer informed the IDE status	**Insufficient General Usability**

Similarly, in Table 26 we map the papers from in our second SLR (Section 2.3 and Table 6) to our subthemes. By aligning these findings with our thematic framework, we ensured a comprehensive and cohesive understanding of the challenges with adoption of these tools.

Table 26: Mapping Found Issues from Second Literature Review to Our Thematic Adoption Framework

Paper	Addressed Adoption Sub-themes
(Holtmann et al., 2024)	Obsolescence, Emergence of Agile Approaches, Lack of Top Management Support, Developer Resistance, Simple Product, Simple Needs, Lack of Skills
(Henderson & Salado, 2024)	Insufficient Utility, Lack of Top Management Support, Lack of Strategy for Change Management, Cost and ROI Uncertainty, Lack of Skills
(Abrahão et al., 2017)	Excess Complexity, Insufficient General Usability, Lack of Skills
(Verbruggen & Snoeck, 2023)	Excess Complexity, Lack of Strategy for Change Management, Cost and ROI Uncertainty
(Escalona et al., 2022)	Insufficient Utility, Developer Resistance, Cost and ROI Uncertainty, Lack of Skills
(Akundi et al., 2022)	Excess Complexity, Lack of Top Management Support, Cost and ROI Uncertainty, Lack of Skills
(Bucchiarone, Cabot, et al., 2020)	Excess Complexity, Insufficient Support, Insufficient General Usability, Security and Confidentiality Concerns, Emergence of Agile Approaches
(Huldt & Stenius, 2019)	Excess Complexity, Lack of Top Management Support, Cost and ROI Uncertainty, Lack of Skills
(Chami & Bruel, 2018)	Lack of Top Management Support, Developer Resistance, Lack of Strategy for Change Management, Cost and ROI Uncertainty
(Akdur et al., 2018)	Lack of Skills
(Giraldo et al., 2018)	Developer Resistance
(Vogelsang et al., 2017)	Excess Complexity, Insufficient Utility, Lack of Top Management Support, Lack of Strategy for Change Management, Insufficient Usability, Cost and ROI Uncertainty
(Whittle et al., 2017)	Insufficient Support, Security and Confidentiality Concerns, Developer Resistance, Cost and ROI Uncertainty, Lack of Skills
(Giraldo et al., 2018)	Lack of Strategy for Change Management
(Morris et al., 2016)	Insufficient Support
(Bonnet et al., 2015)	Excess Complexity, Developer Resistance, Lack of Strategy for Change Management
(Vallecillo, 2015)	Insufficient General Usability, Insufficient Support, Lack of Top Management Support, Developer Resistance, Cost and ROI Uncertainty, Lack of Skills

7.2 Analysis of Themes and Sub-themes Across Different Research Methods

Table 27 presents a detailed analysis of challenges associated with software modeling tools by examining the frequency of mentions across different research methods – literature reviews, case studies, grounded theory, and surveys. The frequencies are calculated once per participant and per sub-theme. While this approach provides insights into which issues were frequently discussed, these frequency numbers should be interpreted with caution. The numbers do not necessarily reflect the severity of the issues raised, many minor concerns may be more frequently mentioned than major ones. Additionally, not all sub-themes were given equal attention, with some potentially over-represented due to specific research focuses. As such, this analysis reflects both the research design choices and the relative prominence of certain sub-themes, rather than their definitive importance.

Table 27: Analysis of Themes and Sub-themes Across Different Research Methods

Theme	Sub-theme	SLR 1	SLR 2	Case-study Survey	Grounded Theory	Survey (Mean)
Theme A: Tooling and Support Concerns (Section 5.4.1)	Sub-theme A1: Excess Complexity	16	6		4	>3
	Sub-theme A2: Insufficient Support	8	3	1	2	>3
	Sub-theme A3: Insufficient Utility	26	2	13	5	>3
	Sub-theme A4: Obsolescence		1		1	>3
	Sub-theme A5: Security and Confidentiality Concerns		2			
	Sub-theme A6: Insufficient Reliability	5		3		
	Sub-theme A6: Insufficient General Usability	11	3	19		
Theme B: Challenges from Modern Work Paradigms (Section 5.4.2)	Sub-theme B1: Emergence of Agile Approaches		2		6	
	Sub-theme B2: Effective Communication Mechanisms				2	
	Sub-theme B3: Advancements in Documentation Techniques				2	
	Sub-theme B4: Rise to Dominance of Standard Frameworks and Architectures				1	
Theme C: Organizational and Behavioural Resistance (Section 5.4.3)	Sub-theme C1: Lack of Top Management Support		7		3	
	Sub-theme C2: Project Manager Resistance				1	
	Sub-theme C3: Developer Resistance		7		3	

Theme	Sub-theme	SLR 1	SLR 2	Case-study Survey	Grounded Theory	Survey (Mean)
	Sub-theme C4: Lack of Strategy for Change Management		6			
Theme D: Lack of Perceived Value (Section 5.4.4)	Sub-theme D1: Dominance of Code				4	>3
	Sub-theme D2: Reduced Efficacy of Complex Models	1			4	
	Sub-theme D3: Simple Product, Simple Needs		1		5	
	Sub-theme D4: Cost and ROI Uncertainty	3	7			
Theme E: Knowledge and Skill Deficiency (Section 5.4.5)	*Sub-theme E1: Lack of Awareness*				3	>3
	Sub-theme E2: Lack of Skills	2	8		2	

Theme A, Tooling and Support Concerns, explores various issues associated with software modeling tools, highlighting significant challenges in complexity, support, utility, obsolescence, security, reliability, and usability. This theme, based on its sub-themes and the frequency of mentions across different research methods, suggests widespread recognition of problems affecting the adoption and effective use of these tools.

With 16 mentions in Literature1 and 6 in Literature2, ***Excess Complexity*** is a well-documented concern, indicating that this issue is recognized both in academic research and industry-related publications. The four mentions in grounded theory further validate that this complexity is a critical issue identified in practical user settings. The survey data showing a mean score of (>3) reinforces that a majority of respondents find complexity a notable barrier. These high counts across multiple methods underscore that simplifying the user interface and functionality of modeling tools is crucial for better adoption.

Subthemes A3 **Insufficient Utility** and A2 **Insufficient Support** show consistent recognition across different research methods. This suggests a gap in the availability or accessibility of utility and support for users, which should be better addressed by tool developers.

Although concerns like obsolescence and security are mentioned less frequently, their presence in survey results suggests these are still relevant issues for certain user groups, emphasizing the need for ongoing updates and targeted security measures.

Theme B, Challenges from Modern Work Paradigms, while noted less frequently among other research methods, emerged consistently with grounded theory, stating how user-generated insights emphasize everyday practical challenges, offering deep, qualitative insights into the issues most directly impacting users. With 6 mentions in grounded theory, the ***Emergence of Agile Approaches*** subtheme suggests that the dynamic and fast-paced nature of modern development environments may conflict with traditional modeling approaches. Although these challenges are less dominant in overall mentions, they point to the evolving nature of work practices and the necessity for modeling tools to adapt to agile and other modern technologies.

Theme C, organizational and behavioral resistance, also features prominently, particularly through grounded theory and literature reviews. For instance, ***Lack of Top Management Support*** and ***Developer Resistance*** are each frequently mentioned (seven times in Literature2 and three in grounded theory), reflecting internal cultural barriers that prevent the successful integration of modeling tools. These mentions indicate that resistance within an organization, whether from top management or developers, significantly affects the adoption and utilization of these tools. This highlights the importance of strategic change management and the need to align organizational culture with the adoption of new technologies.

Theme D, Lack of Perceived Value, is moderately discussed but still significant, particularly through literature and grounded theory. Issues like ***Cost and ROI Uncertainty*** (seven mentions in Literature2) suggest ongoing skepticism about the tangible benefits of modeling tools. The data implies that users often question the value added by these tools, especially when simpler, more direct coding methods are available. This theme's prominence reflects the need for clearer communication of the benefits and value proposition of modeling tools, perhaps through showcasing successful case studies or providing ROI analyses.

Theme E, Knowledge and Skill Deficiency, is another critical area. For example, ***Lack of Skills*** received eight mentions in literature 2, illustrating that knowledge gaps and insufficient training are commonly perceived obstacles. ***Lack of Awareness*** is highlighted by both grounded theory and survey results, indicating a high consensus among a broader population about its significance.

7.3 Mapping Our Findings to the UTAUT Model

Figure 43 illustrates the alignment of various sub-themes identified in our research with the core constructs of the unified theory of acceptance and use of technology, UTAUT (Venkatesh et al., 2003), which is explained in Section 2.1.6.

By aligning our themes and sub-themes with the core constructs of UTAUT – Performance Expectancy, Effort Expectancy, Social Influence, and Facilitating Conditions – we aim to provide a comprehensive understanding of the factors influencing the adoption of software modeling tools. This mapping facilitates the identification of key drivers and barriers, enabling us to propose targeted strategies for improving technology acceptance and usage among developers.

The colors used in the figure are consistent with our color-coding scheme from previous sections. The sub-themes have white backgrounds, while the themes are displayed with a lighter shade of their respective colors. For example, themes like “Perceived Value,” “Aligned with Modern Work Paradigms,” and “Knowledge and Skill” use lighter background shades to distinguish them clearly.

The topic “Perceived Benefits” is an overarching theme that includes sub-themes such as “Better Development,” “Better Communication,” and “Information Representation.” This hierarchical structuring was implemented to optimize space and enhance clarity in the figure.

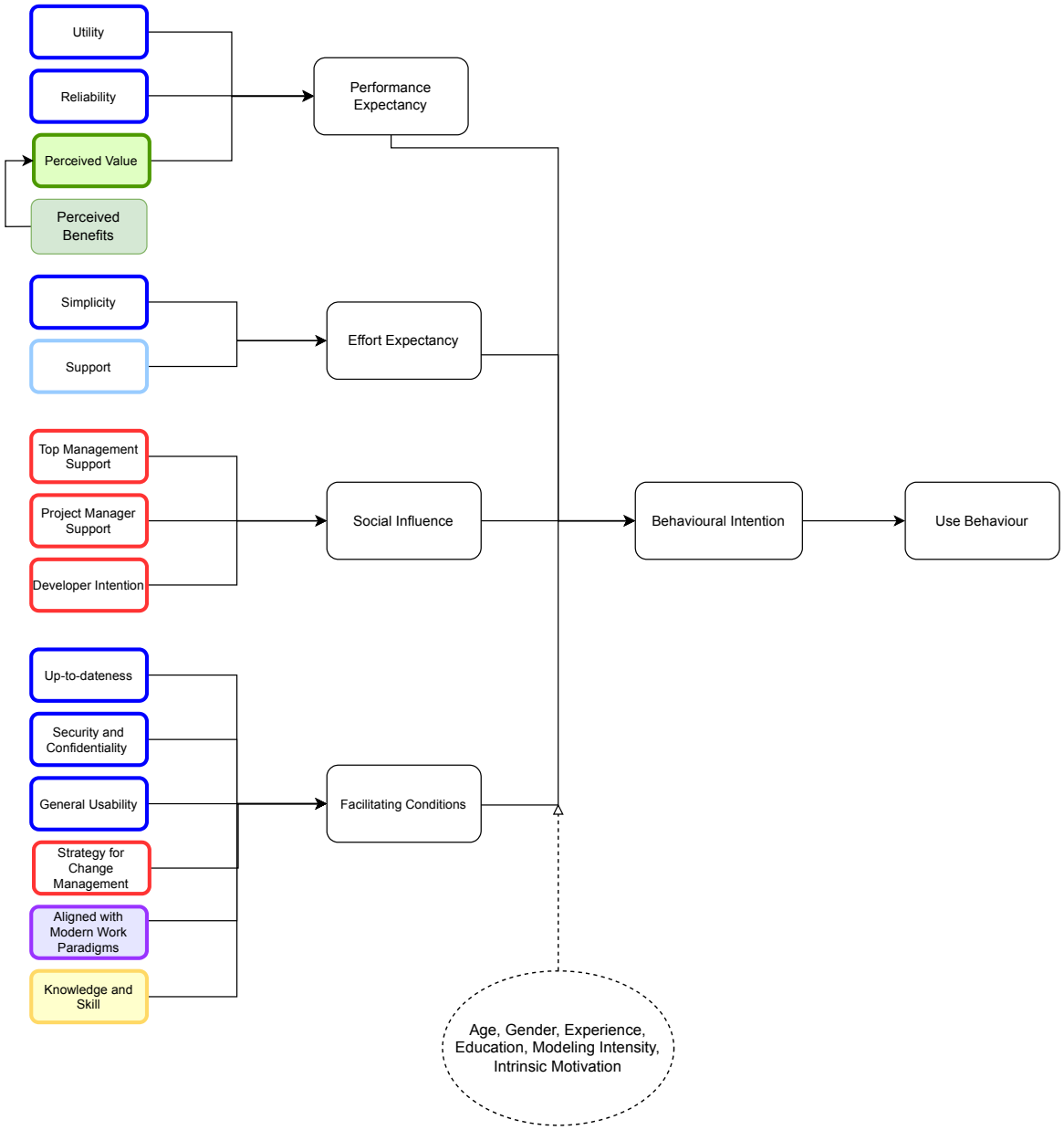


Figure 43: Mapping Findings to the UTAUT Model

7.3.1 UTAUT Construct: Performance Expectancy

Performance Expectancy refers to the degree to which an individual believes that using the technology will help them to attain gains in job performance. The sub-themes and themes related to this construct include:

- Utility
- Reliability
- Perceived Value including sub-themes mentioned in Section 5.4.4.
- Perceived Benefits including themes mentioned in Section 5.5.

7.3.2 UTAUT Construct: Effort Expectancy

Effort Expectancy is the degree of ease associated with the use of the technology. The sub-themes associated with this construct are:

- Simplicity
- Support

7.3.3 UTAUT Construct: Social Influence

Social Influence is the degree to which an individual perceives that important others believe they should use the new system. The relevant sub-themes include:

- Top Management Support
- Project Manager Support
- Developer Intention

7.3.4 UTAUT Construct: Facilitating Conditions

Facilitating Conditions refer to the degree to which an individual believes that an organizational and technical infrastructure exists to support the use of the system. These factors highlight the necessity of a supportive environment, current and reliable tools, security measures, and comprehensive change management strategies to facilitate the adoption of software modeling tools. The themes and sub-themes related to this construct are:

- Up-to-datedness
- Security and Confidentiality
- General Usability
- Strategy for Change Management
- Aligned with Modern Work Paradigms including sub-themes mention in Section 5.4.2.
- Knowledge and Skill including sub-themes mentioned in Section 5.4.5.

7.3.5 UTAUT Construct: Moderating Variables

The figure also incorporates moderating variables such as Age, Gender, Experience, Education, Modeling Intensity, and Intrinsic Motivation. These variables could potentially influence the strength and direction of the relationships between the UTAUT constructs and the Behavioral Intention and Use Behavior of the technology.

7.4 Answers to Research Questions

In this section, we will summarize how the findings of this thesis address each of the research questions.

RQ1: What are the current research practices, evaluation methods, and key trends, findings, and gaps in studying Model-Based Engineering (MBE) tools?

The thesis systematically reviewed the literature (Chapter 2), revealing key trends in MBE tool evaluations, including the prevalent methods and data collection techniques. The analysis highlighted common characteristics, such as the tools and model types frequently used, the participant profiles, and the challenges associated with MBE tools. Notably, the reviews identified gaps, particularly in considering human factors in modeling experience evaluations.

RQ2: What are the human-centric barriers to the adoption of software modeling identified in the literature?

The literature analysis (Section 2.3) uncovered several human-centric barriers, which are summarized in Table 26.

RQ3: What types of classifications of adoption barriers are reported in the literature?

Adoption barriers in MBE are categorized in various ways (Section 2.3.4). Some studies provide general lists of challenges, while others classify issues into specific themes. Common classifications include:

- Organizational factors (knowledge, integration, culture)
- Tool-related issues
- Cultural and political challenges
- Technical obstacles (immature tools, lack of validation)
- Economic concerns (costs, investment fears)

RQ4: How do UmpleOnline users experience flow while using the platform?

Our results (Chapter 4) show that UmpleOnline users experience a moderate level of flow when they are using the tool. Detailed statistics are presented in Table 9 and Table 10.

RQ5: What are the perceived drivers and barriers to using UmpleOnline among its users?

The survey with UmpleOnline users (Chapter 4) identified several drivers, including ease of use, tool flexibility, and support for model-based practices. However, barriers such as limited functionality, and issues with the user interface were also noted. These findings contribute to a better understanding of the factors that influence the adoption and continued use of modeling tools like UmpleOnline.

RQ6: How do developers perceive and use software modeling tools in their daily work?

The grounded theory exploration (Chapter 5) revealed that developers have varied perceptions of modeling tools, influenced by their individual experiences, project requirements, and the specific tools they use. While some developers view modeling tools as essential for design and documentation, others see them as cumbersome, preferring more flexible, less formal methods. This diversity in perception underscores the need for tools that are adaptable to different workflows and user preferences.

RQ7: Is there a relationship between developers' perception of modeling tools and demographic factors such as location, experience, and usage intensity?

The survey data analysis (Chapter 6) found significant correlations between developers' perceptions of modeling tools and factors such as experience level and usage intensity. Experienced developers who use modeling tools extensively tend to have more favorable views, recognizing the benefits these tools offer. However, demographic factors like location did not show a strong influence, suggesting that perceptions are more closely tied to individual and professional contexts than geographical ones.

RQ8: What would be an appropriate conceptual framework that would capture factors contributing to the adoption of software modeling tools?

The thesis proposes a conceptual framework that integrates the empirical findings from the literature review, case study, and grounded theory research. This framework highlights the critical factors influencing the adoption of software modeling tools, including tooling and support, aligning with modern work paradigms, organizational support, perceived value, and

knowledge and skills which are mapped to UTAUT adoption framework and shown in Figure 43. It serves as a foundation for future research and practical applications, offering a structured approach to enhancing the adoption and effective use of modeling tools.

7.5 Summary of Contributions

This thesis presents several key contributions to the understanding and adoption of software modeling tools. These contributions are summarized as follows:

1. **Thematic Taxonomy for Perceived Barriers in Section 5.4**:
    - We developed a detailed taxonomy that categorizes the perceived barriers to adopting software modeling tools. This taxonomy identifies and organizes the specific challenges faced by developers, such as usability issues, lack of support, and organizational resistance.
2. **Thematic Taxonomy for Perceived Benefits in Section 5.5**:
    - We created a comprehensive taxonomy that outlines the perceived benefits of using software modeling tools. This taxonomy highlights the advantages such as improved communication, enhanced development quality, and better information representation, which can be leveraged to encourage adoption and demonstrate value to stakeholders.
3. **A New Taxonomy for Different Types of Software Modeling Tools in Section 5.9**:
    - We developed a new taxonomy that classifies software modeling tools into distinct categories, such as informal tools, diagram-focused tools, analysis and generation capable tools, and maximally formal tools. This classification helps in understanding the unique features, strengths, and limitations of each tool type.
4. **Insights into Current Practices and Trends among Developers in Section 5.10**:
    - We analyzed current practices and emerging trends among software developers in the context of modeling tool usage. This analysis provides a snapshot of the

prevailing attitudes, methodologies, and tools in use, offering a benchmark for future studies and tool development efforts.

5. **Thematic Framework for Software Modeling Adoption in Section 7.3**:
    - We established a thematic framework that integrates the identified barriers and benefits to offer a holistic view of the factors influencing the adoption of software modeling tools. This framework presents an innovative approach compared to previous classifications (Section 2.3.4) by incorporating both technical and organizational factors, alongside social and psychological elements. It expands on traditional adoption barriers by explicitly integrating themes like management support, change management strategies, and alignment with modern work paradigms. Unlike earlier classifications, this model accounts for individual variables such as experience, intrinsic motivation, and modeling intensity.
6. **Empirical Evidence from Multiple Methods**:
    - We utilized a multi-method research approach, including literature reviews, interviews, surveys, and grounded theory studies, to gather empirical evidence supporting the thematic framework. This rigorous methodology promotes the reliability and validity of the findings, offering a well-rounded perspective on the issues at hand. The data is available for others to build on.

By addressing these contributions, this thesis advances both the theoretical understanding and practical application of software modeling tools, offering valuable insights and recommendation for stakeholders involved in software development.

7.6 Threats to Validity and How We Addressed the Threats

In this research, several potential threats to validity were identified and addressed to ensure the robustness and reliability of the findings. These threats span across different aspects

of the research methodology, data collection, and analysis. Here we discuss the primary threats and the measures taken to mitigate them:

7.6.1 Construct Validity:

Misalignment between the theoretical constructs and the actual measurements could lead to inaccurate conclusions. ***Mitigation****: To address this, we utilized established frameworks and validated instruments from the literature whenever possible. In situations where existing instruments were not applicable, we designed the survey with guidance and confirmation from our research supervisor, who is an expert in survey design. Additionally, we conducted pilot studies to refine our survey instruments and interview guides, ensuring they accurately captured the constructs of interest. This approach helped ensure that our data collection tools were both reliable and valid, contributing to the overall rigor of our research.*

7.6.2 Internal Validity:

The presence of confounding variables could influence the results, leading to erroneous interpretations of causal relationships. ***Mitigation****: We used a multi-method approach, and multiple measurements combining qualitative and quantitative data, to triangulate findings and identify potential confounders. Additionally, careful control and documentation of the research process helped minimize the impact of these variables.*

7.6.3 External Validity:

Limited generalizability of the findings due to a non-representative sample. ***Mitigation***: *We aimed for a randomized and diverse sample by including participants from various organizations, industries, and geographical locations. The use of multiple data sources (literature reviews, interviews, surveys) further enhanced the generalizability of the results.*

7.6.4 Reliability:

Inconsistencies in data collection and analysis could compromise the reliability of the findings. ***Mitigation****: We ensured consistency by standardizing data collection procedures and using reliable data analysis techniques. Two researchers were involved in the coding and analysis of qualitative and quantitative data to perform cross-checking and enhance reliability.*

By addressing these threats to validity, we aimed to enhance the credibility and reliability of our research findings, providing a solid foundation for further investigation and practical application in the field of software modeling tool adoption.

7.7 Residual Threats to Validity

Despite implementing various mitigation strategies, several residual threats to validity remain in our study, which highlight the need for caution in interpreting our findings.

- **Search Engine Limitations**: While Google Scholar is commonly used in systematic literature reviews, it may not be the most effective tool for processing Boolean queries, potentially limiting the comprehensiveness of our literature reviews. In other words, papers could have been missed through a combination of Google's lack of Boolean logic, combined with the possibility of there being relevant papers unknown to the authors of both the papers Google found as well as all the snowballed papers.

- **Inclusion Criteria**: Our inclusion criteria restricted the literature reviews to specific publication years, with the second literature review focusing solely on the last ten years. This limitation may have excluded relevant insights from earlier works.

- **Bias in Data Analysis**: There may have been biases in our data analysis, particularly concerning UmpleOnline, which was developed by the supervisor.

- **Classification Challenges**: The study involved multiple classifications, and the boundaries between groups were often subtle and not easily discernible, increasing the risk of misclassification. Furthermore, as tool vendors regularly change their targeted audience and add new features, classifications may become outdated or less applicable over time, potentially leading to further misclassification between groups. Tools may also have not advertised features clearly in their documentation, leading us, for example to not realize that a tool could analyse models or generate code.

- **Recruitment Bias:** The interview invitation was advertised through the social networks and personal contacts of the researcher and her supervisor, which may have introduced bias in participant selection and affected the diversity of responses.

7.8 Future Work

This thesis has provided valuable insights into the adoption of software modeling tools, identifying key themes and sub-themes that influence their acceptance and usage. However, several areas warrant further investigation to build on the findings presented here.

- **Validate Presented Framework**: Further research is needed to test the applicability and robustness of the thematic framework presented in this thesis. The framework, which categorizes the key factors influencing the adoption of software modeling tools, should be validated across diverse organizational contexts and environments to ensure its generalizability and effectiveness.
- **Assess Findings for Various Types of Modeling Tools:** In this thesis, we have presented four types of modeling tools including informal tools, diagram focused tools, analysis and generation capable tools, and maximally formal tools. Given the diverse nature of these tools, it is essential to assess these findings across different types of MBE tools. This approach will

provide a nuanced understanding of the unique challenges and benefits associated with each tool category, allowing for tailored recommendations to improve their adoption and usability.

- **Investigate Human Factors Further:** More research is needed to explore the human factors influencing the adoption and effective use of software modeling tools. Understanding how cognitive, emotional, and social factors impact user interaction with these tools can provide valuable insights for improving their design and implementation.
- **Explore the Role of UX Design:** Further research is essential to explore the impact of improved UX design on user satisfaction and the adoption of software modeling tools. This can be achieved by conducting extensive usability testing and controlled experiments to evaluate various design elements in these tools. Such analyses can lead to the development of comprehensive design guidelines for software modeling tools, which are critical for enhancing usability and promoting widespread adoption.
- **Consider the Impact of Emerging Technologies*:*** As technology continues to advance, it is important to investigate how emerging technologies such as artificial intelligence, and machine learning might influence the adoption and effectiveness of software modeling tools.

7.9 Recommendations

Based on the findings of this research, several recommendations can be made to various stakeholders to improve the adoption and usage of software modeling tools:

7.9.1 Recommendations for Tool Developers:

- **Enhance Usability**: Address issues of excess complexity and insufficient general usability. Focus on creating intuitive interfaces and providing comprehensive user support. Many of the points identified in our case study work (Chapter 4), and our analysis of mindset barriers to modeling (Sectin 5.4) will be relevant.
- **Improve Utility**: Ensure that tools provide the most important features and capabilities, according targeted users' requirements and modern work practices, to enhance user trust and perceived value. Some of the most frequently requested features include robust support for

collaborative work, clear ***separation of concerns***, effective ***versioning*** capabilities, and expanded ***options for diagram types*** and ***code generation across multiple languages***. Integrating and ***supporting modern frameworks and technologies*** effectively within software modeling practices is also crucial for increasing the adoption and utilization of modeling tools.

- **Prioritize Security and Confidentiality**: Prioritize security features to address concerns about data privacy and confidentiality.
- **Address Obsolescence**: Continuously update tools to align with the latest technological advancements while also focusing on building a strong user base and community to ensure ongoing support. Consider making the tools open source to provide long-term accessibility and sustainability. Additionally, ensure backward compatibility so that models created in earlier versions remain usable in later versions, and clearly communicate this feature to users.

7.9.2 Recommendations for Development Managers:

- **Provide Support and Training**: Provide adequate support and training for developers to mitigate knowledge and skill deficiencies. Invest in training programs and resources or select tools that come with their own built-in approaches to training such as interactive user manuals with examples. This does not have to be expensive or deep, since good tools can sell themselves.
- **Encourage Adoption**: Actively promote the benefits of software modeling tools (as we have outlined in Section 5.5) to teams. Highlight success stories and provide incentives for adoption. People who have had experience with good tools could be invited to give webinars demonstrating how the tools have improved productivity and quality.
- **Implement Change Management Strategies**: Implement a clear strategy for change management to address resistance from project managers and developers. Involve stakeholders in the decision-making process to foster a sense of ownership and acceptance.
- **Ensure Top Management Support**: Ensure top management visibly supports and prioritizes the adoption of modeling tools to influence other stakeholders positively.

7.9.3 Recommendations for Developers:

- **Engage in Continuous Learning**: Take advantage of available training resources to improve skills and knowledge related to software modeling tools.
- **Advocate for Support**: Communicate the need for support and resources to development managers and tool developers. Share feedback on tool usability and utility to drive improvements.
- **Collaborate with Peers**: Work collaboratively with peers to share best practices and solutions to common challenges encountered when using software modeling tools.

By addressing the identified challenges and leveraging the recommendations provided, stakeholders can significantly improve the adoption and effective use of software modeling tools, ultimately leading to better software development outcomes.

References

Abrahão, S., Bordeleau, F., Cheng, B., Kokaly, S., Paige, R., Stöerrle, H., & Whittle, J. (2017). User Experience for Model-Driven Engineering: Challenges and Future Directions. *Proceedings of 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, 229–236. https://doi.org/10.1109/MODELS.2017.5

Agarwal, R., & Karahanna, E. (2000). Time Flies When You're Having Fun: Cognitive Absorption and Beliefs about Information Technology Usage. *Journal of MIS Quarterly*, *24*, 665–694. https://doi.org/10.2307/3250951

Agner, L. T. W., & Lethbridge, T. C. (2017). A Survey of Tool Use in Modeling Education. *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, 303–311. https://doi.org/10.1109/MODELS.2017.1

Agner, L. T. W., Lethbridge, T. C., & Soares, I. W. (2019). Student experience with software modeling tools. *Software & Systems Modeling*, *18*(5), 3025–3047. https://doi.org/10.1007/s10270-018-00709-6

Agt-Rickauer, H., Kutsche, R.-D., & Sack, H. (2018). DoMoRe – A Recommender System for Domain Modeling: *Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development*, 71–82. https://doi.org/10.5220/0006555700710082

Ahmar, Y. E., Gérard, S., Dumoulin, C., & Pallec, X. L. (2015). Enhancing the communication value of UML models with graphical layers. *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, 64–69. https://doi.org/10.1109/MODELS.2015.7338236

Akdur, D., Garousi, V., & Demirörs, O. (2018). A survey on modeling and model-driven engineering practices in the embedded software industry. *Journal of Systems Architecture*, *91*, 62–82. https://doi.org/10.1016/j.sysarc.2018.09.007

Akundi, A., Ankobiah, W., Mondragon, O., & Luna, S. (2022). Perceptions and the extent of Model-Based Systems Engineering (MBSE) use – An industry survey. *Proceedings of 2022 IEEE International Systems Conference (SysCon)*, 1–7. https://doi.org/10.1109/SysCon53536.2022.9773894

Almalki, S. (2016). Integrating Quantitative and Qualitative Data in Mixed Methods Research—Challenges and Benefits. *Journal of Education and Learning*, *5*(3), 288–296.

Auer, M., Meyer, L., & Biffl, S. (2007). Explorative UML Modeling—Comparing the Usability of UML Tools. *ICEIS 2007 - 9th International Conference on Enterprise Information Systems*, 466–473.

Badreddin, O., Khandoker, R., Forward, A., Masmali, O., & Lethbridge, T. (2018). A Decade of Software Design and Modeling: A Survey to Uncover Trends of the Practice. *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, 245–255. https://doi.org/10.1145/3239372.3239389

Baltes, S., & Ralph, P. (2021). Sampling in Software Engineering Research: A Critical Review and Guidelines. *Empirical Software Engineering*, *27*(4), 94. https://doi.org/10.1007/s10664-021-10072-8

Beck, K. (1999). *Extreme Programming Explained*. Addison-Wesley Professional. https://www.oreilly.com/library/view/extreme-programming-explained/0201616416/

Bhattacherjee, A. (2012). *Social Science Research: Principles, Methods, and Practices*. https://digitalcommons.usf.edu/oa_textbooks/3

Bonnet, S., Voirin, J.-L., Normand, V., & Exertier, D. (2015). Implementing the MBSE Cultural Change: Organization, Coaching and Lessons Learned. *Journal of INCOSE International Symposium*, *25*, 508–523. https://doi.org/10.1002/j.2334-5837.2015.00078.x

Booch, G., Maksimchuk, R. A., Engle, M. W., Young, B. J., Connallen, J., & Houston, K. A. (2008). *Object-oriented analysis and design with applications, third edition* (Vol. 33). Addison-Wesley Professional. https://dl.acm.org/doi/10.1145/1402521.1413138

Bordeleau, F., Liebel, G., Raschke, A., Stieglbauer, G., & Tichy, M. (2017). Challenges and Research Directions for Successfully Applying MBE Tools in Practice. *MDETOOLS 2017*, 6.

Brambilla, M., Cabot, J., & Wimmer, M. (2017). *Model-Driven Software Engineering in Practice: Second Edition*. Morgan & Claypool Publishers.

Brown, A. (2004). Model driven architecture: Principles and practice. *Journal of Software and System Modeling*, *3*, 314–327. https://doi.org/10.1007/s10270-004-0061-2

Bucchiarone, A., Cabot, J., Paige, R. F., & Pierantonio, A. (2020). Grand challenges in model-driven engineering: An analysis of the state of the research. *Journal of Software and Systems Modeling*, *19*(1), Article 1. https://doi.org/10.1007/s10270-019-00773-6

Bucchiarone, A., Ciccozzi, F., Lambers, L., Pierantonio, A., Tichy, M., Tisi, M., Wortmann, A., & Zaytsev, V. (2021). What Is the Future of Modeling? *Journal of IEEE Software*, *38*, 119–127. https://doi.org/10.1109/MS.2020.3041522

Bucchiarone, A., Savary-Leblanc, M., Pallec, X. L., Bruel, J.-M., Cicchetti, A., Cabot, J., Gerard, S., Aslam, H., Marconi, A., & Perillo, M. (2020). Papyrus for gamers, let's play modeling. *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, 1–5. https://doi.org/10.1145/3417990.3422002

Chami, M., & Bruel, J.-M. (2018). A Survey on MBSE Adoption Challenges. *Proceedings of The Systems Engineering Conference of the Europe*, 1–16. https://oatao.univ-toulouse.fr/22637/

Charmaz, K. (2006). Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis. In *Introducing Qualitative Methods* (Vol. 1).

Cohen, S., & Bodner, E. (2019). The relationship between flow and music performance anxiety amongst professional classical orchestral musicians. *Journal of Psychology of Music*, *47*(3), 420–435. https://doi.org/10.1177/0305735618754689

Condori-Fernández, N., Panach, J. I., Baars, A. I., Vos, T., & Pastor, Ó. (2013). An empirical approach for evaluating the usability of model-driven tools. *Science of Computer Programming*, *78*(11), 2245–2258. Scopus. https://doi.org/10.1016/j.scico.2012.07.017

Corbin, J., & Strauss, A. (2008). *Basics of Qualitative Research (3rd ed.): Techniques and Procedures for Developing Grounded Theory*. SAGE Publications, Inc. https://doi.org/10.4135/9781452230153

Creswell, J. W., & Plano Clark, V. L. (2018). *Designing and Conducting Mixed Methods Research* (Third edition). https://us.sagepub.com/en-us/nam/designing-and-conducting-mixed-methods-research/book241842

Creswell, J. W., & Poth, C. N. (2016). *Qualitative Inquiry and Research Design: Choosing Among Five Approaches* (Fourth Edition). SAGE Publications.

Csikszentmihalyi, M. (2000). *Beyond boredom and anxiety* (pp. 231). Jossey-Bass.

Davis, F. D. (1986). *A technology acceptance model for empirically testing new end-user information systems: Theory and results* [Thesis, Massachusetts Institute of Technology]. https://dspace.mit.edu/handle/1721.1/15192

Dawadi, S., Shrestha, S., & Giri, R. A. (2021). Mixed-Methods Research: A Discussion on its Types, Challenges, and Criticisms. *Journal of Practical Studies in Education*, *2*(2), Article 2.

Dedoose (Version 9.0.107). (n.d.). [Computer software]. Retrieved July 5, 2023, from https://www.dedoose.com/

Dickerson, C. E., & Mavris, D. (2013). A Brief History of Models and Model Based Systems Engineering and the Case for Relational Orientation. *IEEE Systems Journal*, *7*(4), Article 4. IEEE Systems Journal. https://doi.org/10.1109/JSYST.2013.2253034

Egyed, A. (2006). Instant consistency checking for the UML. *Proceeding of the 28th International Conference on Software Engineering - ICSE '06*, 381. https://doi.org/10.1145/1134285.1134339

Eichelberger, H., Eldogan, Y., & Schmid, K. (2009). A Comprehensive Survey of UML Compliance in Current Modelling Tools. *Software Engineering 2009*, 39–50.

El Kouhen, A., Dumoulin, C., Gérard, S., & Boulet, P. (2012). *Evaluation of Modeling Tools Adaptation*. *Journal of HAL Open Science*

Escalona, M. J., de Koch, N. P., & Rossi, G. H. (2022). A Quantitative SWOT-TOWS Analysis for the Adoption of Model-Based Software Engineering. *Journal of Object Technology*. https://doi.org/10.5381/jot.2022.21.4.a9

Ferreira, J. J., de Souza, C. S., & Cerqueira, R. (2014). Characterizing the Tool-notation-people Triplet in Software Modeling Tasks. *IHC'14, Brazilian Symposium on Human Factors in Computing Systems*, 31–40.

Flick, U., Kardoff, E. von, & Steinke, I. (2004). *A Companion to Qualitative Research*. SAGE.

France, R., & Rumpe, B. (2007). Model-driven Development of Complex Software: A Research Roadmap. *Proceedings of Future of Software Engineering (FOSE '07)*, 37–54. https://doi.org/10.1109/FOSE.2007.14

Friedenthal, S., Moore, A., & Steiner, R. (2014). *A practical guide to SysML: The systems modeling language. (3rd ed.)*. Morgan Kaufmann. https://shop.elsevier.com/books/a-practical-guide-to-sysml/friedenthal/978-0-12-800202-5

Giraldo, F. D., España, S., Pastor, Ó., & Giraldo, W. J. (2018). Considerations about quality in model-driven engineering. *Software Quality Journal*, *26*(2), 685–750. https://doi.org/10.1007/s11219-016-9350-6

Hamari, J., & Koivisto, J. (2014). Measuring flow in gamification: Dispositional Flow Scale-2. *Journal of Computers in Human Behavior*, *40*, 133–143. https://doi.org/10.1016/j.chb.2014.07.048

Hartson, R., & Pyla, P. (2019). What Are UX and UX Design? In *The UX Book* (pp. 3–25). Elsevier. https://doi.org/10.1016/B978-0-12-805342-3.00001-1

Hassenzahl, M. (2008). User experience (UX): Towards an experiential perspective on product quality. *Proceedings of IHM '08*, *339*, 11–15. https://doi.org/10.1145/1512714.1512717

Hassenzahl, M., & Tractinsky, N. (2006). User experience—A research agenda. *Behaviour & Information Technology*, *25*(2), 91–97. https://doi.org/10.1080/01449290500330331

Haxthausen, A. E., & Peleska, J. (2016). On the Feasibility of a Unified Modelling and Programming Paradigm. *Proceedings of Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications*, *9953*, 32–49. https://doi-org/10.1007/978-3-319-47169-3_4

Headley, M. G., & Plano Clark, V. L. (2020). Multilevel Mixed Methods Research Designs: Advancing a Refined Definition. *Journal of Mixed Methods Research*, *14*(2), 145–163. https://doi.org/10.1177/1558689819844417

Heena, & Ranjna. (2011). A comparative study of UML tools. *Proceedings of the International Conference on Advances in Computing and Artificial Intelligence*, 1–4. https://doi.org/10.1145/2007052.2007053

Heldal, R., Pelliccione, P., Eliasson, U., Lantz, J., Derehag, J., & Whittle, J. (2016). Descriptive vs prescriptive models in industry. *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, 216–226. https://doi.org/10.1145/2976767.2976808

Henderson, K., McDermott, T., & Salado, A. (2024). MBSE adoption experiences in organizations: Lessons learned. *Systems Engineering*, *27*(1), 214–239. https://doi.org/10.1002/sys.21717

Henderson, K., & Salado, A. (2021). Value and benefits of model-based systems engineering (MBSE): Evidence from the literature. *Journal of Systems Engineering*, *24*(1), 51–66. https://doi.org/10.1002/sys.21566

Henderson, K., & Salado, A. (2024). The Effects of Organizational Structure on MBSE Adoption in Industry: Insights from Practitioners. *Journal of Engineering Management*, *36*(1), 117–143. https://doi.org/10.1080/10429247.2023.2210494

Henry, P. (2015). Rigor in Qualitative research: Promoting quality in Social Science Research. *Journal of Recent Sciences*, *4(IVC-2015)*.

Hettel, T., Lawley, M., & Raymond, K. (2008). Model Synchronisation: Definitions for Round-Trip Engineering. *Proceedings of International Conference on Theory and Practice of Model Transformations*, *5063*. https://doi.org/10.1007/978-3-540-69927-9_3

Holt, J. (2004). *UML for Systems Engineering: Watching the Wheels* (Second edition). IET.

Holtmann, J., Liebel, G., & Steghöfer, J.-P. (2024). Processes, methods, and tools in model-based engineering—A qualitative multiple-case study. *Journal of Systems and Software*, *210*, 111943. https://doi.org/10.1016/j.jss.2023.111943

Hove, S. E., & Anda, B. (2005). Experiences from conducting semi-structured interviews in empirical software engineering research. *Proceedings of 11th IEEE International Software Metrics Symposium (METRICS'05)*, 10 pp. – 23. https://doi.org/10.1109/METRICS.2005.24

Hsieh, H.-F., & Shannon, S. E. (2005). Three Approaches to Qualitative Content Analysis. *Journal of Qualitative Health Research*, *15*(9), 1277–1288. https://doi.org/10.1177/1049732305276687

Huang, K.-H., Nunes, N., Nóbrega, L., Constantine, L., & Chen, M. (2011). *Hammering Models: Designing Usable Modeling Tools*. *6948*, 537–554. https://doi.org/10.1007/978-3-642-23765-2_37

Huang, M.-H. (2003). Designing website attributes to induce experiential encounters. *Journal of Computers in Human Behavior*, *19*(4), 425–442. https://doi.org/10.1016/S0747-5632(02)00080-8

Huldt, T., & Stenius, I. (2019). State-of-practice survey of model-based systems engineering. *Journal of Systems Engineering*, *22*(2), 134–145. https://doi.org/10.1002/sys.21466

Jackson, S., Martin, A., & Eklund, R. (2008). Long and Short Measures of Flow: The Construct Validity of the FSS-2, DFS-2, and New Brief Counterparts. *Journal of Sport & Exercise Psychology*, *30*, 561–587. https://doi.org/10.1123/jsep.30.5.561

Jira | Issue & Project Tracking Software | Atlassian. (n.d.). Retrieved February 14, 2024, from https://www.atlassian.com/software/jira

Kalantari, R., & Lethbridge, T. C. (2022a). Characterizing UX Evaluation in Software Modeling Tools: A Literature Review. *Journal of IEEE Access*, *10*, 131509–131527. IEEE Access. https://doi.org/10.1109/ACCESS.2022.3227504

Kalantari, R., & Lethbridge, T. C. (2022b). Preliminary results of measuring flow experience in a software modeling tool: UmpleOnline. *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, 923–928. https://doi.org/10.1145/3550356.3559099

Kalantari, R., & Lethbridge, T. C. (2023). *Unveiling Developers' Mindset Barriers to Software Modeling Adoption*. *Proceedings of the 26th International Conference on Model Driven*

Engineering Languages and Systems: Companion Proceedings, 737–746. https://doi.org/10.1109/MODELS-C59198.2023.00120

Kitchenham, B. (2004). *Procedures for Performing Systematic Reviews* (No. 0400011T.1; Issue 0400011T.1). Keele University.

Kitchenham, B. A. (1996). Evaluating software engineering methods and tool part 1: The evaluation context and evaluation methods. *ACM SIGSOFT Software Engineering Notes*, *21*(1), 11–14. https://doi.org/10.1145/381790.381795

Kuhn, A., Murphy, G. C., & Thompson, C. A. (2012). An Exploratory Study of Forces and Frictions Affecting Large-Scale Model-Driven Development. In R. B. France, J. Kazmeier, R. Breu, & C. Atkinson (Eds.), *Model Driven Engineering Languages and Systems* (pp. 352–367). Springer. https://doi.org/10.1007/978-3-642-33666-9_23

Kuusinen, K. (2015). Software Developers as Users: Developer Experience of a Cross-Platform Integrated Development Environment. *Product-Focused Software Process Improvement*, 546–552. https://doi.org/10.1007/978-3-319-26844-6_40

Kuusinen, K. (2016). Are Software Developers Just Users of Development Tools? Assessing Developer Experience of a Graphical User Interface Designer. In C. Bogdan, J. Gulliksen, S. Sauer, P. Forbrig, M. Winckler, C. Johnson, P. Palanque, R. Bernhaupt, & F. Kis (Eds.), *Human-Centered and Error-Resilient Systems Development* (pp. 215–233). Springer International Publishing. https://doi.org/10.1007/978-3-319-44902-9_14

Lahtinen, S., & Peltonen, J. (2005). Adding speech recognition support to UML tools. *Journal of Visual Languages & Computing*, *16*(1), 85–118. https://doi.org/10.1016/j.jvlc.2004.08.001

Law, L.-C., Roto, V., Hassenzahl, M., Vermeeren, A., & Kort, J. (2009). *Understanding, scoping and defining user experience: A survey approach*. 719–728. https://doi.org/10.1145/1518701.1518813

Lenberg, P., Feldt, R., & Wallgren, L. G. (2015). Behavioral software engineering. *Journal of Systems and Software*, *107*(C), 15–37. https://doi.org/10.1016/j.jss.2015.04.084

Lethbridge, T. C., Abdelzad, V., Husseini Orabi, M., Husseini Orabi, A., & Adesina, O. (2016). Merging Modeling and Programming Using Umple. In T. Margaria & B. Steffen (Eds.), *Proceedings of Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications* (pp. 187–197). Springer International Publishing. https://doi.org/10.1007/978-3-319-47169-3_14

Lethbridge, T. C., Forward, A., Badreddin, O., Brestovansky, D., Garzon, M., Aljamaan, H., Eid, S., Husseini Orabi, A., Husseini Orabi, M., Abdelzad, V., Adesina, O., Alghamdi, A., Algablan, A., & Zakariapour, A. (2021). Umple: Model-driven development for open source and education. *Journal Od Science of Computer Programming*, *208*, 102665. https://doi.org/10.1016/j.scico.2021.102665

Lewis, J. R., & Sauro, J. (2021). Usability and User Experience: Design and Evaluation. In *Handbook of Human Factors and Ergonomics* (pp. 972–1015). John Wiley & Sons, Ltd. https://doi.org/10.1002/9781119636113.ch38

Liebel, G., Marko, N., Tichy, M., Leitner, A., & Hansson, J. (2018). Model-based engineering in the embedded systems domain: An industrial survey on the state-of-practice. *Software & Systems Modeling*, *17*(1), 91–113. https://doi.org/10.1007/s10270-016-0523-3

List of Unified Modeling Language tools. (2020, October 9). Wikipedia. https://en.wikipedia.org/wiki/List_of_Unified_Modeling_Language_tools

Madni, A. M., & Sievers, M. (2018). Model-based systems engineering: Motivation, current status, and research opportunities. *Jornal of Systems Engineering*, *21*(3), 172–190. https://doi.org/10.1002/sys.21438

Marin, M., & Bhattacharya, J. (2013). Getting into the musical zone: Trait emotional intelligence and amount of practice predict flow in pianists. *Journal of Frontiers in Psychology*, *4*. https://doi.org/10.3389/fpsyg.2013.00853

Méndez Fernández, D., & Passoth, J.-H. (2019). Empirical software engineering: From discipline to interdiscipline. *Journal of Systems and Software*, *148*, 170–179. https://doi.org/10.1016/j.jss.2018.11.019

Mert Ozkaya & Ferhat Erata. (2018). *Analysing UML Modeling Tools for Practical Use*. Retrieved from https://www.semanticscholar.org/paper/Analysing-UML-Modeling-Tools-for-Practical-Use

Morris, B. A., Harvey, D., Robinson, K. P., & Cook, S. C. (2016). Issues in Conceptual Design and MBSE Successes: Insights from the Model-Based Conceptual Design Surveys. *INCOSE International Symposium*, *26*(1), 269–282. https://doi.org/10.1002/j.2334-5837.2016.00159.x

Myers, B. A., Ko, A. J., Latoza, T. D., & Yoon, Y. (2016). Programmers Are Users Too: Human-centered methods for improving programming tools. *Computer*, *49*(7), 44–52. Scopus. https://doi.org/10.1109/MC.2016.200

Nagamachi, M. (Ed.). (2010). *Kansei/Affective Engineering*. CRC Press. 10.1201/EBK1439821336

Neto, V. V. G., Basso, F., Santos, R. P. dos, Bakar, N. H., Kassab, M., Werner, C., Oliveira, T., & Nakagawa, E. Y. (2019). Model-driven engineering ecosystems. *Proceedings of the 7th International Workshop on Software Engineering for Systems-of-Systems and 13th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems*, 58–61. https://doi.org/10.1109/SESoS/WDES.2019.00016

Nielsen, J. (1994, November 1). *Usability Inspection Method Summary: Article by Jakob Nielsen*. Nielsen Norman Group. https://www.nngroup.com/articles/summary-of-usability-inspection-methods/

Norman, D. (1988). *The Design of Everyday Things*. The MIT Press. https://mitpress.mit.edu/9780262640374/the-design-of-everyday-things/

Norman, D. (2004). Emotional Design: Why We Love (or Hate) Everyday Things. New York, Basic Books.

Norman, G. (2010). Likert scales, levels of measurement adn the “laws” of statistics. *Advances in Health Sciences Education : Theory and Practice*, *15*, 625–632. https://doi.org/10.1007/s10459-010-9222-y

Ozkaya, M. (2019). Are the UML modelling tools powerful enough for practitioners? A literature review. *IET Software*, *13*(5), 338–354. https://doi.org/10.1049/iet-sen.2018.5409

Ozkaya, M., & Erata, F. (2020). Understanding Practitioners' Challenges on Software Modeling: A Survey. *Journal of Computer Languages*, *58*, 100963. https://doi.org/10.1016/j.cola.2020.100963

Panach, J. I., Condori-Fernández, N., Baars, A., Vos, T., & Pastor, Ó. (2011). *An Experimental Usability Evaluation Framework for Model-Driven Tools*. *Journal of Membrane Science* - J MEMBRANE SCI. 640-643. https://doi.org/10.1007/978-3-642-23768-3_102.

Papyrus. (2020, December 8). https://www.eclipse.org/papyrus/

Paz, F., & Pow-Sang, J. (2016). A systematic mapping review of usability evaluation methods for software development process. *International Journal of Software Engineering and Its Applications*, *10*, 165–178. https://doi.org/10.14257/ijseia.2016.10.1.16

Peifer, C., Wolters, G., Harmat, L., Heutte, J., Tan, J., Freire, T., Tavares, D., Fonte, C., Andersen, F., Hout, J., Pola, L., Ceja, L., & Triberti, S. (2022). A Scoping Review of Flow Research. *Journal of Frontiers in Psychology*, *13*. https://doi.org/10.3389/fpsyg.2022.815665

Petre, M. (2013). UML in practice. *2013 35th International Conference on Software Engineering (ICSE)*, 722–731. https://doi.org/10.1109/ICSE.2013.6606618

Pietron, J., Raschke, A., Stegmaier, M., Tichy, M., & Rukzio, E. (2018). A study design template for identifying usability issues in graphical modeling tools. *2nd Workshop on Tools for Model Driven Engineering at MODELS'18*, *2245*, 336–345. Scopus.

Planas, E., & Cabot, J. (2020). How are UML class diagrams built in practice? A usability study of two UML tools: Magicdraw and Papyrus. *Computer Standards & Interfaces*, *67*, 1–13. https://doi.org/10.1016/j.csi.2019.103363

Pourali, P. (2018). An Empirical Investigation to Understand the Dificulties and Challenges of Sofware Modellers When Using Modelling Tools. *Proceedings of ACM/IEEE 21th International Conference on Model Driven Engineering Languages and Systems (MODELS '18)*, 224–234. https://doi.org/10.1145/3239372.3239400

Rau, P.-L., Peng, S.-Y., & Yang, C.-C. (2006). Time Distortion for Expert and Novice Online Game Players. *Journal of Cyberpsychology & Behavior : The Impact of the Internet, Multimedia and Virtual Reality on Behavior and Society*, *9*, 396–403. https://doi.org/10.1089/cpb.2006.9.396

Ren, R., Castro, J. W., Santos, A., Pérez-Soler, S., Acuña, S. T., & de Lara, J. (2020). Collaborative Modelling: Chatbots or On-Line Tools? An Experimental Study. *Proceedings of the Evaluation and Assessment in Software Engineering*, 260–269. https://doi.org/10.1145/3383219.3383246

Rheinberg, F., & Engeser, S. (2018). Intrinsic Motivation and Flow. In J. Heckhausen & H. Heckhausen (Eds.), *Motivation and Action* (pp. 579–622). Springer International Publishing. https://doi.org/10.1007/978-3-319-65094-4_14

Ribeiro, A., Sousa, L. de, & Silva, A. R. da. (2016). Comparative analysis of workbenches to support DSMLs: Discussion with non-trivial Model-Driven Development needs. *2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, 323–330. https://doi.org/10.5220/0005745603230330

Robles Luna, E., Sánchez Begines, J. M., Rivero, J. M., Morales, L., Enríquez, J. G., & Rossi, G. H. (2018). Challenges for the adoption of model-driven web engineering approaches in

industry. *Journal of Web Engineering*, *17, nos. 3–4*, 183–205. https://doi.org/10.5555/3370055.3370057

Runeson, P., & Höst, M. (2008). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131. https://doi.org/10.1007/s10664-008-9102-8

Runeson, P., Rainer, A., Höst, M., & Regnell, B. (2012). Design of the Case Study. In *Case Study Research in Software Engineering* (pp. 23–45). John Wiley & Sons, Ltd. https://doi.org/10.1002/9781118181034.ch3

Ryan, R. M. (1982). Control and information in the intrapersonal sphere: An extension of cognitive evaluation theory. *Journal of Personality and Social Psychology*, *43*(3), 450–461. https://doi.org/10.1037/0022-3514.43.3.450

Saariluoma, P., & Jokinen, J. P. P. (2014). Emotional Dimensions of User Experience: A User Psychological Analysis. *International Journal of Human–Computer Interaction*, *30*(4), 303–320. https://doi.org/10.1080/10447318.2013.858460

Safdar, S. A., Iqbal, M. Z., & Khan, M. U. (2015). Empirical Evaluation of UML Modeling Tools–A Controlled Experiment. In G. Taentzer & F. Bordeleau (Eds.), *Modelling Foundations and Applications* (pp. 33–44). Springer International Publishing. https://doi.org/10.1007/978-3-319-21151-0_3

Saraiva, J., & Silva, A. (2008). Evaluation of MDE Tools from a Metamodeling Perspective. *Journal of Database Management*, *19*, 21–46. https://doi.org/10.4018/jdm.2008100102

Saunders, M., Lewis, P., & Thornhill, A. (2019). *Research Methods for Business Students: Vol. 8th Edition*. Pearson. https://www.pearson.com/uk/educators/higher-education-

educators/program/Saunders-Research-Methods-for-Business-Students-7th-Edition/PGM1089011.html

Savary-Leblanc, M. (2019). *Improving MBSE Tools UX with AI-Empowered Software Assistants—IEEE Conference Publication*. 648–652. https://doi.org/10.1109/MODELS-C.2019.00099

Savary-Leblanc, M., Le Pallec, X., & Gérard, S. (2024). Understanding the need for assistance in software modeling: Interviews with experts. *Journal of Software and Systems Modeling*, *23*(1), 103–135. https://doi.org/10.1007/s10270-023-01104-6

Seaman, C. B. (1999). Qualitative methods in empirical studies of software engineering. *Journal of IEEE Transactions on Software Engineering*, *25*(4), 557–572. IEEE Transactions on Software Engineering. https://doi.org/10.1109/32.799955

Seffah, A., & Rilling, J. (2001). Investigating the relationship between usability and conceptual gaps for human-centric CASE tools. *IEEE Symposium on Human Centric Computing Languages and Environments*, 226–231. https://doi.org/10.1109/HCC.2001.995263

Selic, B. (2016). Programming ⊂ Modeling ⊂ Engineering. *Proceedings of Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications.*, *9953*, 11–26. https://doi.org/10.1007/978-3-319-47169-3_2

Sharp, H., Dittrich, Y., & de Souza, C. R. B. (2016). The Role of Ethnographic Studies in Empirical Software Engineering. *Journal of IEEE Transactions on Software Engineering*, *42*(8), 786–804. https://doi.org/10.1109/TSE.2016.2519887

Siegl, D. (2015). *Why Round-Trip Engineering does not work*. https://blog.lieberlieber.com/2015/09/05/why-round-trip-engineering-does-not-work/

Stegmaier, M., Raschke, A., Tichy, M., Meßner, E., Hajian, S., & Feldengut, A. (2019). Insights for Improving Diagram Editing Gained from an Empirical Study. *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 405–412. https://doi.org/10.1109/MODELS-C.2019.00063

Störrle, H. (2007). Large scale modeling efforts: A survey on challenges and best practices. *Proceedings of the 25th Conference on IASTED International Multi-Conference*, 382–389.

Störrle, H. (2018). Improving model usability and utility by layered diagrams. *Proceedings of the 10th International Workshop on Modelling in Software Engineering*, 59–66. https://doi.org/10.1145/3193954.3193958

Strauss, A., & Corbin, J. M. (1990). *Basics of qualitative research: Grounded theory procedures and techniques* (p. 270). Sage Publications, Inc.

Vallecillo, A. (2015). On the Industrial Adoption of Model Driven Engineering. Is your company ready for MDE? *Journal of Information Systems and Software Engineering for Big Companies*, *1*(1), 52–68.

Venkatesh, V., Morris, M., Davis, G., & Davis, F. (2003). User Acceptance of Information Technology: Toward a Unified View. *Journal of MIS Quarterly*, *27*, 425–478. https://doi.org/10.2307/30036540

Verbruggen, C., & Snoeck, M. (2023). Practitioners' experiences with model-driven engineering: A meta-review. *Journal of Software and Systems Modeling*, *22*(1), 111–129. https://doi.org/10.1007/s10270-022-01020-1

Vogelsang, A., Amorim, T., Pudlitz, F., Gersing, P., & Philipps, J. (2017). Should I Stay or Should I Go? On Forces that Drive and Prevent MBSE Adoption in the Embedded Systems Industry. In M. Felderer, D. Méndez Fernández, B. Turhan, M. Kalinowski, F. Sarro, & D. Winkler (Eds.), *Product-Focused Software Process Improvement* (pp. 182–198). Springer International Publishing. https://doi.org/10.1007/978-3-319-69926-4_14

Wang, X., Conboy, K., & Cawley, O. (2012). "Leagile" software development: An experience report analysis of the application of lean approaches in agile software development. *Journal of Systems and Software*, *85*(6), 1287–1299. https://doi.org/10.1016/j.jss.2012.01.061

Weber, T., Zoitl, A., & HuBmann, H. (2019). Usability of Development Tools: A CASE-Study. *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 228–235. https://doi.org/10.1109/MODELS-C.2019.00037

Webster, J., Trevino, L. K., & Ryan, L. (1993). The dimensionality and correlates of flow in human-computer interactions. *Journal of Computers in Human Behavior*, *9*(4), 411–426. https://doi.org/10.1016/0747-5632(93)90032-N

Whittle, J., Hutchinson, J., Rouncefield, M., Burden, H., & Heldal, R. (2013). Industrial adoption of model-driven engineering: Are the tools really the problem? *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8107 LNCS*, 1–17. Scopus. https://doi.org/10.1007/978-3-642-41533-3_1

Whittle, J., Hutchinson, J., Rouncefield, M., Burden, H., & Heldal, R. (2017). A taxonomy of tool-related issues affecting the adoption of model-driven engineering. *Software and Systems Modeling*, *16*(2), 313–331. Scopus. https://doi.org/10.1007/s10270-015-0487-8

Williams, J., Matragkas, N., Kolovos, D., Ananiadou, S., & Paige, R. (2014). Software Analytics for MDE Communities. *Proceedings of the 1st Workshop on Open Source Software for Model Driven Engineering (OSS4MDE'14) Co-Located with ACM/IEEE 17th International Conference on Model Driven Engineering Languages & Systems (MoDELS 2014)*, 1–10.

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14*, 1–10. https://doi.org/10.1145/2601248.2601268

Wohlin, C., & Aurum, A. (2015). Towards a decision-making structure for selecting a research design in empirical software engineering. *Journal of Empirical Software Engineering*, *20*(6), 1427–1455. https://doi.org/10.1007/s10664-014-9319-7

Woszczynski, A. B., Roth, P. L., & Segars, A. H. (2002). Exploring the theoretical foundations of playfulness in computer interactions. *Journal of Computers in Human Behavior*, *18*(4), 369–388. https://doi.org/10.1016/S0747-5632(01)00058-9

Wróbel, M. (2013). Emotions in the software development process. *Proceedings of 6th International Conference on Human System Interactions, HSI 2013*, 518–523. https://doi.org/10.1109/HSI.2013.6577875

Zarour, M. (2020). A rigorous user needs experience evaluation method based on software quality standards. *Journal of Telecommunication Computing Electronics and Control*, *18*(5), 2787–2799. https://doi.org/10.12928/telkomnika.v18i5.16061

Zarour, M., & Alharbi, M. (2017). User experience framework that combines aspects, dimensions, and measurement methods. *Journal of Cogent Engineering*, *4*(1), 1421006. https://doi.org/10.1080/23311916.2017.1421006

Zhang, P., & Finneran, C. (2005). Flow in Computer-Mediated Environments: Promises and Challenges. *Journal of Communications of the Association for Information Systems*, *15*, 82–101. https://doi.org/10.17705/1cais.01504

Appendix 1: Flow Survey Ethical Certificate

17/04/2023

Université d'Ottawa
Bureau d'éthique et d'intégrité de la recherche

University of Ottawa
Office of Research Ethics and Integrity

CERTIFICAT D'APPROBATION ÉTHIQUE | CERTIFICATE OF ETHICS APPROVAL

Numéro du dossier / Ethics File Number	H-04-21-6689
Titre du projet / Project Title	Flow Experience in a Software Modeling tool: Umple
Type de projet / Project Type	Thèse de doctorat / Doctoral thesis
Statut du projet / Project Status	Renouvelé / Renewed
Date d'approbation (jj/mm/aaaa) / Approval Date (dd/mm/yyyy)	12/05/2021
Date d'expiration (jj/mm/aaaa) / Expiry Date (dd/mm/yyyy)	11/05/2024

Équipe de recherche / Research Team

Chercheur / Researcher	Affiliation	Role
Reyhaneh KALANTARI	École de science informatique et de génie électrique / School of Electrical Engineering and Computer Science	Chercheur Principal / Principal Investigator
Timothy LETHBRIDGE	École de science informatique et de génie électrique / School of Electrical Engineering and Computer Science	Superviseur / Supervisor

Conditions spéciales ou commentaires / Special conditions or comments

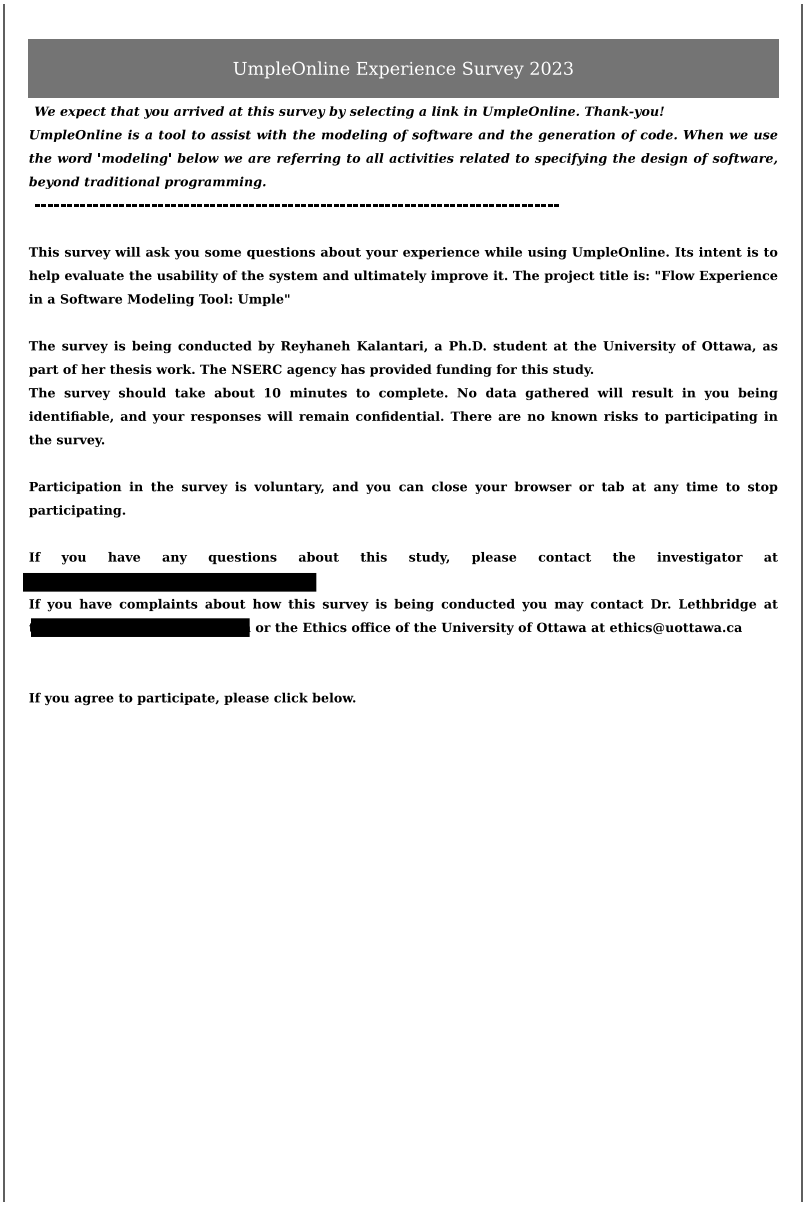
550, rue Cumberland, pièce 154
Ottawa (Ontario) K1N 6N5 Canada

550 Cumberland Street, Room 154
Ottawa, Ontario K1N 6N5 Canada

613-562-5387 • 613-562-5338 • ethique@uOttawa.ca / ethics@uOttawa.ca
www.recherche.uottawa.ca/deontologie | www.recherche.uottawa.ca/ethics

Appendix 2: Flow Survey Questions

The following are the consent form and questions we asked our survey participants in the flow survey. The results are discussed in Chapter 4.



UmpleOnline Experience Survey 2023

We expect that you arrived at this survey by selecting a link in UmpleOnline. Thank-you!
UmpleOnline is a tool to assist with the modeling of software and the generation of code. When we use the word 'modeling' below we are referring to all activities related to specifying the design of software, beyond traditional programming.

This survey will ask you some questions about your experience while using UmpleOnline. Its intent is to help evaluate the usability of the system and ultimately improve it. The project title is: "Flow Experience in a Software Modeling Tool: Umple"

The survey is being conducted by Reyhaneh Kalantari, a Ph.D. student at the University of Ottawa, as part of her thesis work. The NSERC agency has provided funding for this study.
The survey should take about 10 minutes to complete. No data gathered will result in you being identifiable, and your responses will remain confidential. There are no known risks to participating in the survey.

Participation in the survey is voluntary, and you can close your browser or tab at any time to stop participating.

If you have any questions about this study, please contact the investigator at [redacted]
If you have complaints about how this survey is being conducted you may contact Dr. Lethbridge at [redacted] or the Ethics office of the University of Ottawa at ethics@uottawa.ca

If you agree to participate, please click below.

UmpleOnline Experience Survey 2023

* 1. You are: (select all that apply)

- [ ] A grade school student
- [ ] A undergraduate student in a university or college
- [ ] A graduate student (Masters/PhD)
- [ ] A university educator
- [ ] A practitioner in industry
- [ ] Other (please specify)

2. In what country do you currently reside?

Other (please specify)

* 3. How would you describe your experience level in modeling?

- ◯ Inexperienced (I have only created a few very simple models or have created models only in academic courses)
- ◯ Medium
- ◯ Expert (I have been using modeling tools extensively in professional practice for at least 6 months)

* 4. Which modeling tools have you used to create or edit models (please select all that apply)?

- [ ] Papyrus
- [ ] Umple
- [ ] MagicDraw
- [ ] Argo UML
- [ ] Visual Paradigm
- [ ] Other (please specify)
- [ ] Enterprise Architect
- [ ] Astah
- [ ] USE
- [ ] Eclipse based modeling tools

* 5. Why have you selected Umple today (select all that apply)?

- [ ] It is required by my professor/supervisor/teammates
- [ ] It is the only modeling tool I am familiar with
- [ ] I needed one or more particular features that seem only available, or are best in Umple
- [ ] I found it by following a link from another site or a search and decided I would use it
- [ ] It is my preferred tool for software modeling
- [ ] Other (please specify)

* 6. Which Umple tools have you used (select all that apply)?

- [ ] UmpleOnline
- [ ] IDE Umple Plugins for Eclipse, VSCode, etc.
- [ ] Command line Umple compiler

* 7. Which one of the items below best describes the main task you use Umple for?

- ◯ Drawing diagrams
- ◯ Generating a small functioning system
- ◯ Building a large, complex application
- ◯ Analyzing a model for errors or weaknesses
- ◯ Other (please specify)

Please indicate how often (from never to always) you experience the following while you are modeling with UmpleOnline

8. I feel I am competent enough to meet the high demands of the situation.

◯ Never ◯ Rarely ◯ Sometimes ◯ Usually ◯ Always

9. I do things spontaneously and automatically without having to think.

◯ Never ◯ Rarely ◯ Sometimes ◯ Usually ◯ Always

10. I have a strong sense of what I want to do.

◯ Never ◯ Rarely ◯ Sometimes ◯ Usually ◯ Always

11. I have a good idea while I am performing about how well I am doing.

◯ Never ◯ Rarely ◯ Sometimes ◯ Usually ◯ Always

12. I am completely focused on the task at hand.

◯ Never ◯ Rarely ◯ Sometimes ◯ Usually ◯ Always

13. I have a feeling of total control.

◯ Never ◯ Rarely ◯ Sometimes ◯ Usually ◯ Always

14. I am not worried about what others may be thinking of me.

◯ Never ◯ Rarely ◯ Sometimes ◯ Usually ◯ Always

15. The way time passes seems to be different from normal.

◯ Never ◯ Rarely ◯ Sometimes ◯ Usually ◯ Always

16. The experience is extremely rewarding.

◯ Never ◯ Rarely ◯ Sometimes ◯ Usually ◯ Always

For each of the following statements, please indicate how much agree or disagree you are.

* 17. I enjoy using Umple very much.

◯ Strongly disagree ◯ Disagree ◯ Neither agree nor disagree ◯ Agree ◯ Strongly agree

* 18. I am pretty skilled in using Umple.

◯ Strongly disagree ◯ Disagree ◯ Neither agree nor disagree ◯ Agree ◯ Strongly agree

* 19. I use Umple because I have no choice.

◯ Strongly disagree ◯ Disagree ◯ Neither agree nor disagree ◯ Agree ◯ Strongly agree

* 20. It is important to me to do well while using Umple.

◯ Strongly disagree ◯ Disagree ◯ Neither agree nor disagree ◯ Agree ◯ Strongly agree

21. In your opinion, what are the best qualities of Umple?

22. How could Umple better support your work?

Thank you for your participation

Appendix 3: Interview Ethical Certificate

17/11/2023

Université d'Ottawa
Bureau d'éthique et d'intégrité de la recherche

University of Ottawa
Office of Research Ethics and Integrity

CERTIFICAT D'APPROBATION ÉTHIQUE | CERTIFICATE OF ETHICS APPROVAL

Numéro du dossier / Ethics File Number	H-11-23-9872
Titre du projet / Project Title	Developers' Mindsets Regarding Software Modeling Practices
Type de projet / Project Type	Thèse de doctorat / Doctoral thesis
Statut du projet / Project Status	Approuvé / Approved
Date d'approbation (jj/mm/aaaa) / Approval Date (dd/mm/yyyy)	17/11/2023
Date d'expiration (jj/mm/aaaa) / Expiry Date (dd/mm/yyyy)	16/11/2024

Équipe de recherche / Research Team

Chercheur / Researcher	Affiliation	Role
Reyhaneh KALANTARI	École de science informatique et de génie électrique / School of Electrical Engineering and Computer Science	Chercheur Principal / Principal Investigator
Timothy LETHBRIDGE	École de science informatique et de génie électrique / School of Electrical Engineering and Computer Science	Superviseur / Supervisor

Conditions spéciales ou commentaires / Special conditions or comments

550, rue Cumberland, pièce 154
Ottawa (Ontario) K1N 6N5 Canada

550 Cumberland Street, Room 154
Ottawa, Ontario K1N 6N5 Canada

613-562-5387 • 613-562-5338 • ethique@uOttawa.ca / ethics@uOttawa.ca
www.recherche.uottawa.ca/deontologie I www.recherche.uottawa.ca/ethics

Appendix 4: Interview Protocol

The following is the main interview protocol. It was carefully designed to cover different situations and participant types—those using software modeling practices and those who aren't. This protocol served as a guide for researchers, helping them ask personalized questions during each interview. The results are discussed in Chapter 5.

Note that the interviewers will ask follow-up questions related to these, in order to delve into interesting subject matter that may arise, and may not ask every question in every interview.

A. Screening Questions (from recruitment text, but may need to be repeated if not answered)

In what **country** are you located?

How many **years of development experience** do you have?

Please describe in a couple of words the **domains of software development** where you have the most experience.

In a few words, please indicate how you usually **communicate your software design with other software developers, clients and other stakeholders?** *[If they mention UML, class diagrams, Entity-Relation, State Machines, other diagrams, or textual modeling notation,. then they will be classified as using notations]*

What, if any, tools do you use to represent or communicate the high-level design of your system? *[If they mention any tool other than a simple drawing tool, then we will classify them as using modeling tools]*

B. Interview Session

REMEMBER TO TURN ON ZOOM RECORDING

a) … introduce ourselves, and confirm their name and their experience level.

We are going to ask you some question today about **your perception of how you represent and communicate your design**.

a) Are you OK if we record this session? [if no, then will have to email consent form and rely on notes].

b) We **MAY** show you on the screen an informed consent form [share screen]. Please confirm **orally that you provide you consent to this process**?

Thank-you

[unshare screen]

Ok, let's get started.

Clarify with the respondent that we **will talk about tools for abstract representation of design.**

C. If they are using modeling tools:

Which tools are you using currently for modeling? Which ones have you used previously?

When did you start using these tools in development? How long have you been using these tools?

What are the reasons/motivations behind using these tools?

What were the requirements or criteria you employed for selecting these tools?

What kinds of notations and diagrams do you use for development? Why?

In which development stages do you use the modeling tools?

For which tasks are you using these tools most?

Which features of these tools are critical for you/your team?

To what extent are the models (whether or not created by the tools) being used?

Is there any formal process in your organization to use these models/tools? If yes, pleae elaborate on that.

How did these models help in the development process?

How effective are the processes in which you use the current modeling notations/tools?

How could these processes be improved?

How often do you update your models? Are they kept up to date?

What do you like about the modeling tools/notations you use? To what extent do these tools/notations support your needs?

What do you dislike about current modeling tools/notations? (What problems do they have?) What improvements would you like to see?

D. If they are NOT using modeling tools, NOR notations:

If somebody need to figure out the structure of the data in your system, how do they do this?

if they say that they use a tool to create a diagram/special notation, then go back to questions on modeling tools – Part C)

if they say they simply draw on a whiteboard or using a drawing tool, then go to questions on notations for those not using modeling tools – Part E)

if they say coding and comments on the code is enough: ask: Have you experienced any situation where you had difficulty understanding the code and wished there was a tool or diagram that would help explain it?

If somebody need to figure out the relationships among various components in your system, how do they do this?

If somebody need to figure out the behaviour of complex parts of your system, how do they do this?

Have you considered using a specialized tool to help with these?

[if yes] What tools? What have been the barriers to employ them?

[if no] Why not?

Skip to end

E. If they are NOT using modeling tools/ but ARE using notations:

Have you considered using a specialized tool to help with creating and analysing these/this notations?

[if yes] Why did you not adopt a tool? what did you find was missing? what difficulties did you see in adopting the tool?

[if no] Why not?

Go to Part B, question 8.

At the end of session:

1. Is there anything else you would like to add?
2. Is there anyone else you think we would benefit from talking to?

Thank you for your time.

Appendix 5: Final Survey Ethical Certificate

07/06/2023

Université d'Ottawa
Bureau d'éthique et d'intégrité de la recherche

University of Ottawa
Office of Research Ethics and Integrity

CERTIFICAT D'APPROBATION ÉTHIQUE | CERTIFICATE OF ETHICS APPROVAL

Numéro du dossier / Ethics File Number	H-06-22-8172
Titre du projet / Project Title	Developers' perspectives on software modeling practices and tools
Type de projet / Project Type	Thèse de doctorat / Doctoral thesis
Statut du projet / Project Status	Renouvelé / Renewed
Date d'approbation (jj/mm/aaaa) / Approval Date (dd/mm/yyyy)	18/06/2022
Date d'expiration (jj/mm/aaaa) / Expiry Date (dd/mm/yyyy)	17/06/2024

Équipe de recherche / Research Team

Chercheur / Researcher	Affiliation	Role
Reyhaneh KALANTARI	École de science informatique et de génie électrique / School of Electrical Engineering and Computer Science	Chercheur Principal / Principal Investigator
Timothy LETHBRIDGE	École de science informatique et de génie électrique / School of Electrical Engineering and Computer Science	Superviseur / Supervisor

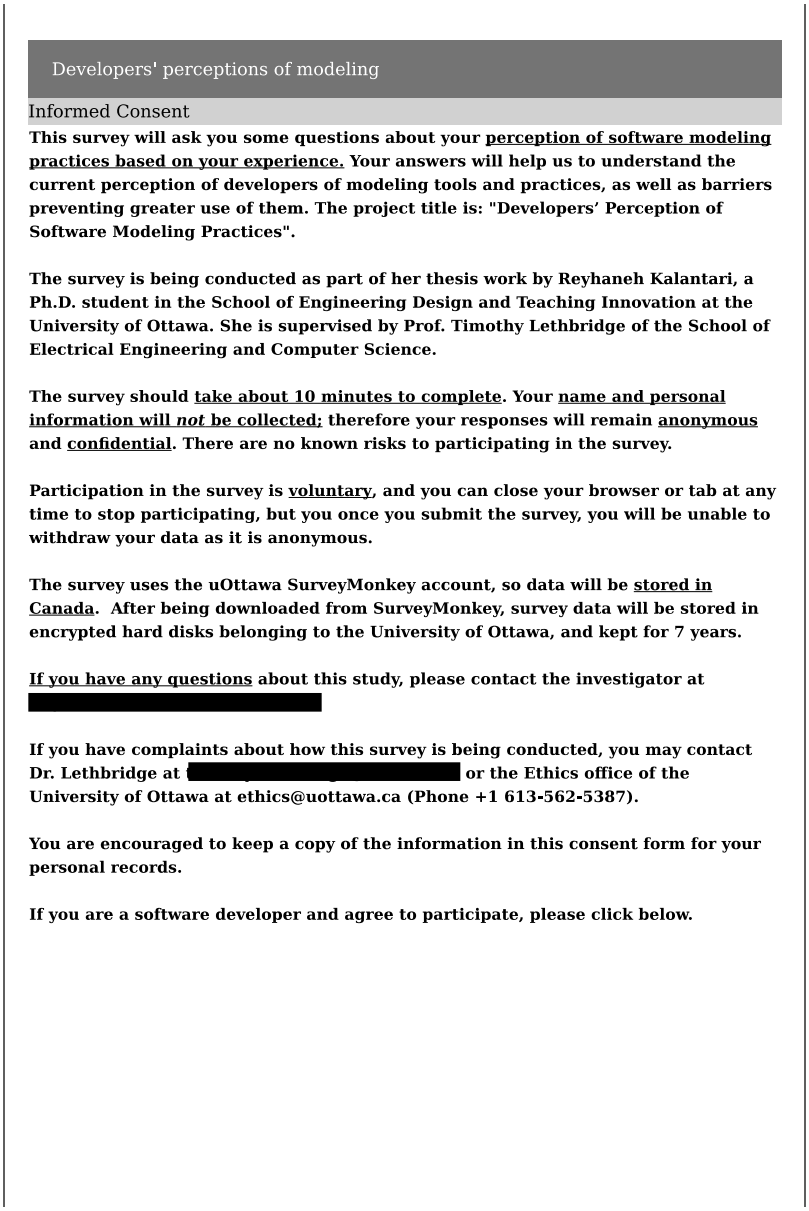
Conditions spéciales ou commentaires / Special conditions or comments

550, rue Cumberland, pièce 154 550 Cumberland Street, Room 154
Ottawa (Ontario) K1N 6N5 Canada Ottawa, Ontario K1N 6N5 Canada

613-562-5387 • 613-562-5338 • ethique@uOttawa.ca / ethics@uOttawa.ca
www.recherche.uottawa.ca/deontologie I www.recherche.uottawa.ca/ethics

Appendix 6: Final Survey Questions

The following are the consent form and questions we asked our survey participants in the final survey. The results are discussed in Chapter 6.



Developers' perceptions of modeling

Informed Consent

This survey will ask you some questions about your perception of software modeling practices based on your experience. Your answers will help us to understand the current perception of developers of modeling tools and practices, as well as barriers preventing greater use of them. The project title is: "Developers' Perception of Software Modeling Practices".

The survey is being conducted as part of her thesis work by Reyhaneh Kalantari, a Ph.D. student in the School of Engineering Design and Teaching Innovation at the University of Ottawa. She is supervised by Prof. Timothy Lethbridge of the School of Electrical Engineering and Computer Science.

The survey should take about 10 minutes to complete. Your name and personal information will *not* be collected; therefore your responses will remain anonymous and confidential. There are no known risks to participating in the survey.

Participation in the survey is voluntary, and you can close your browser or tab at any time to stop participating, but you once you submit the survey, you will be unable to withdraw your data as it is anonymous.

The survey uses the uOttawa SurveyMonkey account, so data will be stored in Canada. After being downloaded from SurveyMonkey, survey data will be stored in encrypted hard disks belonging to the University of Ottawa, and kept for 7 years.

If you have any questions about this study, please contact the investigator at [redacted]

If you have complaints about how this survey is being conducted, you may contact Dr. Lethbridge at [redacted] or the Ethics office of the University of Ottawa at ethics@uottawa.ca (Phone +1 613-562-5387).

You are encouraged to keep a copy of the information in this consent form for your personal records.

If you are a software developer and agree to participate, please click below.

Developers' perceptions of modeling

In this survey we are going to ask you about your personal experiences with software modeling.

We will use the term **modeling** to describe any situation where you are creating or editing a representation of the software that is more abstract than code written in programming languages.

Models may be in the form of *diagrams* (e.g. class diagrams, state diagrams) or *text* (e.g. Dockerfiles, YAML files, Textual UML variants, Formal methods language files). Models be in specialist modeling tools, general drawing tools, even just on whiteboards or paper.

Please answer the questions thinking of **work you have done yourself over the last year.**

1. How would you describe your software modeling intensity in your own development process?

- ◯ Non-modeler (You do not use modeling as described above)
- ◯ Minimal modeler: (You use modeling sporadically, for specific tasks or problem solving)
- ◯ Selective modeler: (You use modeling somewhat consistently, but only for a limited set of tasks)
- ◯ Modeling enthusiast: (You use modeling extensively as an integral part of your development process)

2. How many years of software development experience do you have?

- ◯ Less than 5 years
- ◯ 5-15 years
- ◯ 16- 25 years
- ◯ More than 26 years

3. Thinking of your own experiences with modeling over the last year, please indicate your **level of agreement** with the following statements. You can use the 'Neutral' when you either don't know enough to be able to agree or disagree, or else are partly in agreement and partly in disagreement.

Remember by **modeling,** we mean *any* situation where you are **creating or editing a representation of the software that is more abstract than code** written in programming languages.

	Strongly agree	Agree	Neutral	Disagree	Strongly disagree
Modeling can substantially **reduce costs** by **helping stakeholders to agree** on requirements or high-level designs before writing or changing code.	◯	◯	◯	◯	◯
Models allow **easier communication** with stakeholders					

because they are easier for **non-technical** people to understand than code.	◯	◯	◯	◯	◯
Models allow **quick** communication of **high-level concepts**, hiding details that people may not need to know or understand, or which have not yet been finalized.	◯	◯	◯	◯	◯
Models allow easier communication because they are much more **compact visually** than code.	◯	◯	◯	◯	◯
Models in diagram form are a much **richer communication mode** than code because they allow representation using two dimensions, colours, a variety of shapes, and icons	◯	◯	◯	◯	◯
Modeling can help **prevent defects** as it can make the consequences of requirements or designs clearer.	◯	◯	◯	◯	◯
Developing models is **much less bug-prone** than developing code.	◯	◯	◯	◯	◯
Models allow representation of **domain concepts** or **abstract concepts** that would be too complex to represent concisely in code.	◯	◯	◯	◯	◯
Modeling tools can effectively **analyse** models and point out **potential errors.**	◯	◯	◯	◯	◯
Models allow developers to see different **perspectives or views** of software.	◯	◯	◯	◯	◯
Modeling tools can significantly **reduce**					

the amount of code that has to be written, because code can be **generated**, or models can be **executed at run time**.	○	○	○	○	○
Models can be transformed into **lots of end products**, including code, other models, documentation, user help, user interfaces, etc.	○	○	○	○	○
Modeling languages that give **graphical** views of a system make the system **easier to understand.**	○	○	○	○	○
Modeling **tools** are excessively **complex** and hence challenging to use.	○	○	○	○	○
Modeling **languages** are excessively **complex** and hence challenging to use	○	○	○	○	○
Modeling tools are too **buggy**	○	○	○	○	○
There are not enough good online **examples** of models (of the type and size that would help me to understand how to use languages and tools)	○	○	○	○	○
There is insufficient **documentation** and help about modeling tools and languages.	○	○	○	○	○
Modeling tools and languages **do not have the capabilities** or features I would like.	○	○	○	○	○
I am concerned about the potential **obsolescence** of modeling tools and languages.	○	○	○	○	○
Modeling tools and languages **change too often**, so	○	○	○	○	○

developers have to keep **re-learning** how to use them.	○	○	○	○	○
As tool versions change, older models become **incompatible** with the new versions	○	○	○	○	○
Modeling is *not* very useful in an **agile** software development process.	○	○	○	○	○
Modeling is *not* very useful in a workflow where I can **easily communicate** with other developers through communication mechanisms like Github comments, instant messaging, collaborative editing, and video conferencing.	○	○	○	○	○
Modeling is *not* very useful in **modern workflows** where there are tools for **automatically generating** documentation (e.g. of APIs) and diagrams from code.	○	○	○	○	○
Modeling is **not** very useful when **modern frameworks and technologies** (such as Node, Django, React, Angular, Spring, Flask, Rails, Vue, Docker etc.) mostly define the architecture of systems.	○	○	○	○	○
Good modeling tools are too **expensive**.	○	○	○	○	○
Top management in my organization **discourages** or prevents development teams from using modeling.	○	○	○	○	○
Project managers in my organization are **resistant** to letting their teams do	○	○	○	○	○

modeling.					
Managers in my organization **won't cover the cost** of modeling tools.	◯	◯	◯	◯	◯
Pressure is on me to **deliver working code**; creating or updating models would slow me down.	◯	◯	◯	◯	◯
The **code** is the **most critical artifact**; models are just overhead.	◯	◯	◯	◯	◯
Well-designed **code** is **easy enough for me to understand**, therefore there is little benefit in creating models too.	◯	◯	◯	◯	◯
The best **tools for drawing models** *don't* have features for *generating code*, therefore once the code is written it is challenging or not useful to keep the model up to date.	◯	◯	◯	◯	◯
It is too hard to keep **models and code in synch** and up to date with each other, therefore it is better to just focus on the code.	◯	◯	◯	◯	◯
Code generators that create code from models are **not good enough** for practical use.	◯	◯	◯	◯	◯
As a software **system becomes complex**, any model of that system will get complex too, therefore it will become **less and less useful**.	◯	◯	◯	◯	◯
Software models are **just as complex** as the code, so it is better to work on the code.	◯	◯	◯	◯	◯
Most software projects are **too simple** to benefit	◯	◯	◯	◯	◯

from modeling.					
It would **not be economical** to do modeling in the projects that I work on.	○	○	○	○	○
Most people in the software industry are **not aware** of the capabilities and benefits of modeling.	○	○	○	○	○
I do **not believe that the benefits** of modeling claimed by some people are true.	○	○	○	○	○
It would be **difficult to improve** my modeling skills.	○	○	○	○	○
As a developer, I feel I **don't want to invest any effort** in learning modeling tools.	○	○	○	○	○
As a developer, I feel it would be a **waste of time to increase** the amount of modeling I do.	○	○	○	○	○

4. If you are reluctant to adopt modeling tools, due to other factors not mentioned above, please describe them below.

5. Which modeling tools/methods have you used to create or edit models in the last year (please select all that apply)?

- [ ] Informal Modeling (Whiteboard/paper)
- [ ] Umple
- [ ] ArgoUML
- [ ] Papyrus
- [ ] LucidChart
- [ ] Umlet
- [ ] USE
- [ ] Gliffy
- [ ] Archi /Archimate
- [ ] Diagrams.net /draw.io
- [ ] MagicDraw
- [ ] Rational Software Architect/Designer
- [ ] HCL tools
- [ ] PlantUML
- [ ] Astah
- [ ] Formal methods languages and their tools (Alloy, NuXmV) etc.
- [ ] Other (please specify)

6. In what country do you currently reside?

7. Please describe in a couple of words the domains of software development where you have the most experience. (E.g: websites, aerospace embedded systems, games, database systems, telecom systems, etc.)

8. What is your gender identity?

- ◯ Male
- ◯ Female
- ◯ Non-binary
- ◯ Prefer not to say